Das BASIC-Programmiersystem für DCP

Das vorliegende Kapitel beschreibt kurz eine BASIC-Implementierung. Sie wurde im VEB Robotron-Projekt Dresden für das Betriebssystem DCP geschaffen, das unter anderem auf dem Rechner A 7150 zur Verfügung steht.
Zunächst wird gezeigt, welcher Sprachumfang realisiert wurde und welche Abweichungen und Erweiterungen zum Standard existieren. Dann erhält der Leser einen Überblick über den Aufbau des Systems und die Arbeitsmöglichkeiten.

BASIC für DCP

Die Sprache BASIC für das Betriebssystem DCP realisiert den ANSI-Standard X3.113 (1987).
In der ersten Ausbaustufe des Systems stehen folgende Ausdrucksmittel zur Verfügung:

- Der Sprachkern mit Ausnahme aller MAT-Anweisungen,
- die Editor-Erweiterung,
- eine Integer-Erweiterung für die Arbeit mit ganzzahligen numerischen Größen.

Über die Ausdrucksmittel des Sprachkerns muß hier nichts gesagt werden, da diese Ausdrucksmittel bereits ausführlich beschrieben wurden.
Ein Ziel der Implementation war es, möglichst keine Abweichungen von den Festlegungen des Standards vorzunehmen. Diese Zielstellung konnte bis auf die folgenden Ausnahmen realisiert werden:

Der Standard schreibt vor, daß die Satzlänge des Kanals 0 für die Ein- und Ausgaben über Terminal nicht begrenzt ist. Die Realisierung dieser Forderung hätte in der vorliegenden Implementierung zu einer sehr uneffektiven Lösung geführt. Aus diesem Grunde wurde festgelegt, daß die Satzlänge des Kanals 0 maximal 132 Zeichen beträgt. Für praktische Zwecke dürfte diese Satzlänge völlig ausreichend sein.

- Es sind mehr Schlüsselworte reserviert, als es der Standard vorschreibt.

- Bei der SELECT CASE-Anweisung wird nicht überprüft, ob sich die Konstanten und Wertebereiche in den Auswahllisten überschneiden.

- ist die Spurverfolgung eingeschaltet, so werden bei Wertzuweisungen nur die Zeilennummer, die Indizes und der neue Wert der Variablen ausgegeben.

Wird eine externe Funktion gerufen, aber nicht definiert, so wird nur eine Warnung ausgegeben. Es besteht die Möglichkeit, solche Funktionen als zusätzliche Standardfunktionen in das System einzubinden. Werden in Stringkonstanten unzulässige Zeichen verwendet, so wird ebenfalls nur eine Warnung ausgegeben. Im übrigen können alle Ausdrucksmittel des Sprachkerns (mit Ausnahme der MAT-Anweisungen) so benutzt werden, wie sie im Standard definiert sind.
Die in der Editor-Erweiterung beschriebenen Kommandos sind Bestandteil des Kommandovorrats des Programmiersystems. Sie werden im folgenden Abschnitt behandelt. Erwähnt sei nur, daß neben den vom Standard geforderten Kommandos weitere Kommandos realisiert wurden.
Eine der wesentlichsten Festlegungen des Standards bezüglich der Arbeit mit numerischen Größen besteht darin, daß innerhalb einer Programmeinheit alle numerischen Größen vom gleichen Typ (Gleitkommazahlen doppelter Genauigkeit, Gleitkommazahlen einfacher Genauigkeit oder Festkommazahlen) sind.

Viele andere BASIC-Versionen und andere höhere, problemorientierte Programmiersprachen lassen dagegen in einem Programm bzw. einer Programmeinheit gleichzeitig verschiedene numerische Datentypen zu. Insbesondere ist es für eine effektive Realisierung vieler Algorithmen vorteilhaft, mit ganzzahligen Werten (integer-Werten) zu arbeiten. Zum Beispiel müssen die Indizes von indizierten Variablen stets ganzzahlig sein. Hier ist es besser, von vornherein mit ganzzahligen Werten und nicht mit gerundeten Gleitkommazahlen zu arbeiten. Aus diesen Gründen wurde entschieden, dem Nutzer eine Möglichkeit zur Arbeit mit Integer-Werten zu bieten. Es sei darauf hingewiesen, daß es der Standard durchaus zuläßt, die DECLARE-Anweisung so zu erweitern, daß Variable zusätzlicher Datentypen deklariert werden können. Ein solches Vorgehen hat jedoch drei Nachteile:

-Soll ein Programm, das derartige Erweiterungen benutzt, in ein standardgerechtes Programm, daß heißt ein Programm, das genau den Festlegungen des Standards entspricht, überführt werden, so müssen alle DECLARE-Anweisungen überprüft und gegebenenfalls korrigiert werden. Diese Anweisungen treten an mehreren Stellen im Programm auf.

-Es ist nicht geklärt, wie Konstanten in diesen zusätzlichen Datenarten dargestellt werden. Ist zum Beispiel der Datentyp INTEGER zugelassen, so kann der Prozessor nicht entscheiden, ob eine im Programmtext auftretende Konstante 10 eine ganzzahlige oder eine Gleitkommakonstante sein soll.

-Es besteht keine Möglichkeit, Funktionen zu definieren, die als Resultat einen Wert der zusätzlichen Datentypen liefern. Zwar kann durch eine DECLARE-Anweisung festgelegt werden, daß eine Funktion einen solchen Wert liefern soll. Bei der Definition einer Funktion steht jedoch ein solches Ausdrucksmittel nicht zur Verfügung, wenn nicht eine Erweiterung der syntaktischen Regeln zur Funktionsdefinition vorgenommen wird.

Aus diesen Gründen wurde bei BASIC für DCP ein anderer Weg beschritten. Ziel war es, die Ausdrucksmittel zur Schaffung des zusätzlichen Datentyps INTEGER an einer Stelle im Programm zu konzentrieren und auch die Definition von Funktionen mit Integer-Resultat zuzulassen. Die Konzentration an einer Stelle hat den Vorteil, daß die Überführung in ein standardgerechtes Programm sehr einfach ist.

Will der Nutzer mit Integer-Größen arbeiten, so muß er vor dem Hauptprogramm ein Vorspiel angeben:

```
PRELUDE
prelude-anw
END PRELUDE
```

Die Schlüsselworte PRELUDE und END PRELUDE stehen auf einer gesonderten Programmzeile, das heißt auf einer mit einer Zeilennummer versehenen Zeile. Nach diesen Schlüsselworten darf nur noch ein Kommentar folgen. prelude-anw bezeichnet eine Folge von Anweisungen, die nur im Vorspiel auftreten dürfen und die jeweils auf einer logischen Programmzeile stehen. Da das Vorspiel ausschließlich Informationen für den Prozessor liefert, sind die Anweisungen des Vorspiels keine abarbeitbaren Anweisungen. Zu ihnen darf also nicht verzweigt werden, was auch schon deshalb nicht möglich ist, da das Vorspiel nicht Bestandteil einer Programmeinheit ist.

Im Vorspiel sind zwei Arten, von OPTION-Anweisungen möglich. Die erste hat dieses Aussehen:

```
OPTIO INTEGER liste-num-Ident
```

Dabei bezeichnet liste-num-Ident eine Liste von numerischen Identifikatoren, die durch Kommas getrennt werden. Durch eine OPTION INTEGER-Anweisung wird zweierlei erreicht:

Erstens wird dem Prozessor mitgeteilt, daß im vorliegenden Programm mit Integer-Werten gearbeitet werden soll. Integer-Werte sind ganzzahlige Werte im Bereich von -32768 bis +32767. Alle im Programm auftretenden ganzzahligen Konstanten, die in diesem Bereich liegen, werden intern als Integer-Werte und nicht als Gleitkomma-Werte dargestellt. Zweitens wird den in der Liste aufgeführten Indentifikatoren die Eigenschaft verliehen, im folgenden Programm ausschließlich Integer-Objekte zu bezeichnen. Wird ein solcher Identifikator zur Bezeichnung einer einfachen Variablen verwendet, so kann diese Variable nur Integer-Werte enthalten. Wird er zur Bezeichnung eines Feldes verwendet, so können alle Feldelemente nur Integer-Werte enthalten. Bei der Bezeichnung einer Funktion liefert diese Funktion einen Integer-Wert als Resultat. Durch die OPTION INTEGER-Anweisung werden also keine Objekte definiert. Es wird nur die erwähnte Eigenschaft vermittelt.

Beispiel

```
100 PRELUDE
110 OPTION INTEGER MATRIX,I,J
120 END PRELUDE
200 PROGRAM MATRIXBERECHNUNG
210 DIM MATRIX(20,20)
220 FOR I=1 TO 20
230 FOR J =1 TO 20
240 LET MATRIX(I,J)=1
250 NEXT J
260 NEXT I
170 LET S=O
```

In dem vorliegenden Programmstück werden das Feld MATRIX und die einfachen Variablen I, J und S verwendet. Aufgrund des Vorspiels enthalten das Feld MATRIX und die Variablen I und J Integer-Werte. Die auftretenden Konstanten 0, 1 und 20 sind Integer-Konstanten. Durch die geschachtelten FOR-Anweisungen werden alle Feldelemente mit dem Integer-Wert 1 initialisiert.
Wird die Möglichkeit zur Arbeit mit Integer-Größen genutzt, so können innerhalb einer Programmeinheit numerische Größen als Integer-Größen und Gleitkomma- bzw. Festkommagrößen auftreten. Es müssen besondere Regeln zu ihrer Anpassung festgelegt werden. BASIC für DCP folgt hier der üblichen Praxis der problemorientierten Programmiersprachen:

1. Sind beide Operanden einer dyadischen, arithmetischen Operation Integer-Werte, so ist auch das Ergebnis ein Integer-Wert. Bei der Berechnung wird eine nicht triviale Ausnahme ausgelöst, wenn das Resultat den Bereich der Integer-Werte verläßt (Integer-Überlauf). Die Division zweier Integer-Werte ist eine ganzzahlige Division.
Ist nur ein Operand ein Integer-Wert, so wird er in die Gleitkomma- bzw. Festkommadarstellung konvertiert und die Operation in diesem Format ausgeführt.

2. Muß an eine Integer-Variable ein Gleitkomma- bzw. Festkommawert zugewiesen werden, so wird dieser Wert gerundet und anschließend in

einen Integer-Wert konvertiert. Dabei kann eine nicht triviale Ausnahme (Integer-Überlauf) entstehen.

3. Muß an eine Gleitkomma- bzw. Festkommavariable ein Integer-Wert zugewiesen werden, so wird der Integer-Wert in das geforderte Format konvertiert.

4. Enthält eine numerische LET-Anweisung mehrere Zielvariable, so müssen alle den gleichen Typ besitzen.

Die Anweisung

```
LET A, I, K1 = 20. 5
```

ist also nur korrekt, falls A, 1 und K1 entweder Variable des Typs Integer oder des Typs Gleitkomma bzw. Festkomma sind. Haben sie den Typ Integer, so wird ihnen der Wert 21 zugeordnet.

5. Bei der Übergabe von numerischen Werten an Funktionen und Prozeduren werden die erforderlichen Konvertierungen automatisch vorgenommen, wenn die Parameterübergabe "per Wert" erfolgt. Erfolgt die Parameterübergabe "per Referenz", so müssen der aktuelle und der formale Parameter den gleichen numerischen Typ besitzen.
Das Beispiel MATRIXBERECHNUNG zeigt einen der wichtigsten Anwendungsfälle für Integer-Werte, nämlich als Indizes von indizierten Variablen. Würde das Vorspiel fehlen, so müßten alle Operationen mit den Laufvariablen 1 und j (Zuweisung des Anfangswertes, Erhöhen um die Schrittweite, Endetest, Berechnung der Adresse der indizierten Variablen) im Gleitkommaformat mit anschließender Rundung erfolgen. Damit ist klar, daß die Abarbeitung der Laufanweisungen wesentlich effektiver erfolgt, wenn die Laufvariablen Integer-Größen sind. In diesem Zusammenhang tritt jedoch noch ein weiteres Problem auf:
Die bei der Deklaration von Feldern anzugebenden lndexgrenzen müssen ganzzahlig sein, sind aber in ihrer Größe nicht beschränkt. Das heißt daß die beiden folgenden Felder jeweils 10 Elemente besitzen:

```
DIM A(I TO 10), X(100001 TO 100010)
```

Während für das Feld A die lndexgrenzen Integer-Werte sind, müssen die lndexgrenzen
für das Feld X intern durch Gleitkommazahlen dargestellt werden. Da A durch Redimen-
sionierung zur Laufzeit des Programms Indexgrenzen annehmen kann, die wie bei X nicht durch Integer-Werte dargestellt werden können, müssen die Indexgrenzen generell intern durch Gleitkommazahlen dargestellt werden. Hier liegt die Quelle für eine weitere Uneffektivität, da für jede indizierte Variable zur Programmlaufzeit geprüft wird, ob jeder Index innerhalb der lndexgrenzen liegt. Dieser Vergleich muß im Gleitkommaformat erfolgen, da die lndexgrenzen Gleitkommazahlen sind. In der Mehrzahl der praktischen Fälle könnten lndexgrenzen jedoch als Integer-Werte dargestellt werden; demzufolge ist der erwähnte Test im Integer-Format möglich. Um dies zu ermöglichen, kann im Vorspiel eine weitere OPTION-Anweisung angegeben werden:

```
OPTION BOUNDS [NOT] INTEGER
```

Wird im Vorspiel die Anweisung OPTION BOUNDS INTEGER angegeben, müssen alle Felder, die in den folgenden Programmeinheiten definiert werden, lndexgrenzen besitzen, die im Bereich der Integer-Werte (-32768 bis +32767) liegen. Auch bei Redimensionierungen müssen die neu eingestellten lndexgrenzen in diesem Bereich liegen. Wird eine

Indexgrenze verwendet, die außerhalb dieses Bereiches liegt, so wird ein Fehler gemeldet oder eine Ausnahme ausgelöst.
Ist im Vorspiel die Anweisung OPTION BOUNDS NOT INTEGER angegeben, können bei der Definition und Redimensionierung von Feldern beliebige lndexgrenzen angegeben werden. Diese Anweisung kann auch entfallen, da sie der Standardannahme entspricht.

Das Vorspiel realisiert eine effektive Verarbeitung von Feldern, wenn die Anweisung OPTION BOUNDS INTEGER angegeben und gleichzeitig durch eine OPTION INTEGER-Anweisung die Möglichkeit geschaffen wurde, Integer-Variable als Indizes für indizierte Variable zu verwenden. Im folgenden Beispiel wird dies demonstriert:

Beispiel:

Über Terminal werden zwei Vektoren elementweise eingegeben.
Anschließend wird das
Skalarprodukt gebildet und ausgegeben:

```
10 PRELUDE
20 OPTION INTEGER ANZAHL,I,N
30 OPTION BOUNDS INTEGER
40 END PRELUDE
100 PROGRAM SKALARPRODUKT
110 OPTION BASE 1
120 DIM X(100), Y(100)
200 SUB EINGABE(VEKTOR,ANZAHL)
210 FOR I=LBOUND(VEKTOR) TO LBOUND(VEKTOR)+ANZAHL
220 INPUT PROMPT STR$(I)&". ZAHL = „: VEKTOR(I)
230 NEXT I
240 END SUB
300 INPUT PROMPT "ANZAHL DER ELEMENTE EINES VEKTORS = „: N
310 IF 1 <= N AND N <= 100 THEN 340
320 PRINT "ANZAHL ZU GROSS"
330 STOP
340 PRINT "EINGABE VEKTOR X"
350 CALL EINGABE(X,N)
360 PRINT "EINGABE VEKTOR Y"
370 CALL EINGABE(Y,N)
380 LET SKALARPRODUKT = 0.0
390 FOR I=1 TO N
400 LET SKALARPRODUKT = SKALARPRODUKT+X(I)*Y(I)
410 NEXT I
420 PRINT "SKALARPRODUKT   =     „:SKALARPRODUKT
430 END
```

Dieses Programm kann in einfacher Weise in ein standardgerechtes Programm überführt werden. Es muß lediglich das Vorspiel (die Zeilen mit den Zeilennummern 10 bis 40) gestrichen werden.

Aufbau und Bedienung des BASIC-Programmiersystem für DCP

Zur sicheren Beherrschung des BASIC-Programmiersystems sind einige Kenntnisse über dessen Struktur und Arbeitsweise erforderlich, die hier kurz vermittelt werden.

Systembestandteile

Der gewachsene Sprachumfang von Standard-BASIC erfordert neue Methoden bei der Verarbeitung von BASIC-Programmen. Bisher gab es bei der Arbeit mit BASIC-Textinterpretern zwei Arbeitsphasen: die Programmherstellung bzw. Modifizierung und unmittelbar anschließend den Programmlauf, bei dem eine verdichtete Form des Quelltextes abgearbeitet wurde.
Bei Standard-BASIC kommt zwischen Programmherstellung und Ausführung als weitere Arbeitsphase die Übersetzung hinzu. Bei der Übersetzung wird das Programm auf syntaktische und semantische Fehler untersucht, und es wird ein Zwischencode (RPN-Code) erzeugt, den der Interpreter des Systems abarbeiten kann. Die Übersetzung wird vom BASIC-Programmiersystem automatisch durchgeführt, wenn ein Programm nach Neueingabe oder Änderung erstmals ausgeführt werden soll.

Dateiname des Bestandteils	Funktionen
BAS. EXE	BAS. EXE ist das Kommandoprogramm. Dieses Programm führt die Kommunikation mit dem Benutzer, erkennt die eingegebenen Kommandos und BASIC-Zeilen und aktiviert dementsprechend die übrigen Bestandteile des Systems.
BED. OVR	BED. OVR enthält den Editorteil des Systems. In diesem Teil sind alle Funktionen für die Eingabe, die Korrektur und die Ausgabe des Programmtextes enthalten.
BTR.OVR BP2.OVR	Diese beiden Bestandteile werden zur Übersetzung von BASIC-Programmen in den ausführbaren ' Zwischencode verwendet. BTR.OVR prüft das Programm auf syntaktische und semantische Richtigkeit, und 8P2.OVR erzeugt das ausführbare Zwischenprogramm, wenn kein Fehler erkannt wurde.
BRU.OVR	Dieser Teil enthält den Interpreter, der das vom Übersetzer gelieferte Zwischenprogramm ausführt.

Jede Arbeitsphase wird von bestimmten Bestandteilen des Systems übernommen, die über ein Kommandoprogramm aktiviert werden und die über einen gemeinsamen Datenspeicher Informationen austauschen. Beim Start des BASIC-Systems wird das Kommandoprogramm BAS.EXE aktiviert. Dieses bleibt während der Arbeit mit dem System ständig im Speicher und lädt die OVR-Bestandteile bei bedarf nach. Die OVR-Bestandteile überlagern einander. Sie dürfen vom Benutzer nicht wie ein selbstständiges Programm gestartet werden.

Ein zügiges Arbeiten mit dem System setzt voraus, daß die OVR-Bestandteile möglichst schnell in den Speicher geladen werden können. Die beste Voraussetzung bietet dafür die virtuelle Speicherdiskette. Wird sie verwendet, ergiebt sich die Speicheraufteilung wie in der folgenden Tabelle gezeigt. Die virtuelle Speicherdiskette schränkt dabei den BASIC-Arbeitsspeicher ein Sie sollte daher so dimensioniert werden:

Betriebssystem DCP
Kommandoprogramm BAS.EXE
Speicher für den jeweils benutzen OVR-Teil
BASIC-Arbeitsspeicher. Er enthält: -das Quellprogramm -das zugehörige ausführbare Programm - Übersetzertabellen
Virtuelle Speicherdiskette mit BAS.EXE und den OVR-Teilen

daß sie gerade zur Speicherung der Systembestandteile ausreicht. Wenn auf die virtuelle Speicherdiskette verzichtet werden muß, kann eine Festplatte zur Speicherung der Systembestandteile günstig verwendet werden.

Betreten, Verlassen und Restart des Systems

Das BASIC-System wird beim Start des Kommandoprozessors betreten. Das Gerät, das BAS. EXE und die OVR-Teile enthält, wird als aktuelles Gerät ausgewählt.

Beispiel Die Bestandteile des BASIC-Systems befinden sich auf der virtuellen Speicherdiskette E:. BASIC kann dann durch die folgenden Kommandos gestartet werden:

```
E:
BAS
```

Das BASIC-System meldet sich mit der Ausschrift

Robotron Standard-BASIC, Version xx.xx DCP, 1988

und fordert mit der Eingabeaufforderung

IDENTIFICATION:

einen Identifikationscode an. Als Identifikationscode ist ein Identifikator einzugeben, der mit einem Buchstaben beginnen muß. Nach dem Buchstaben können bis zu fünf Buchstaben oder Ziffern folgen; weitere Ziffern oder Buchstaben sind nicht signifikant. Das Unterstreichungszeichen darf im Identifikationscode nicht verwendet werden.

Der Identifikationscode erfüllt folgende Funktionen:

- Er wird als Dateiname bei OLD-, SAVE- und PROTOCOL-Kommandos verwendet, wenn in
  einem dieser Kommandos der Dateiname nicht explizit angegeben wird .
- Er ist Identifikator des Hauptprogramms, wenn dies keine PROGRAM-Anweisung enthält.
  In diesem Fall darf er nicht zur Bezeichnung anderer Objekte im Programm verwendet werden.
- Er wird als Dateiname verwendet, wenn der BASIC-Arbeitsspeicher in eine Datei gerettet wird.

Es ist empfehlenswert, für die Quelldatei eines BASIC-Programms, für den Programmnamen in der PROGRAM-Anweisung und für den Identifikationscode den gleichen Identifikator zu benutzen.

Nach Eingabe des Identifikationscodes fordert BASIC mit der Eingabeaufforderung
 =>
Kommandos oder Programmzeilen an. Die Arbeit kann dann beginnen.

Beispiel:

Das BASIC-System wird vom Diskettenlaufwerk A: auf die virtuelle Speicherdiskette E: übernommen. Es soll das Programm SORT bearbeitet werden. Das Betriebssystem sei mit der Kommandoaufforderung "/>" generiert. Es ergibt sich folgender Dialog:

```
/>E:
>COPY A: -*.OVR
/>COPY A: BAS.EXE
/>BAS

Robotron Standard-BASIC, Version xx.xx DCP, 1988

IDENTIFICATION: SORT
```

Zum Verlassen des BASIC-Systems wird das BYE-Kommando verwendet. Es hat das Format:

BYE [/-DE]

Die Unterstreichung des Zeichens B im Kommando bedeutet, daß das Schlüsselwort BYE bis auf B abgekürzt werden kann. Zulässig sind also B, BY oder BYE. Anstelle der Großbuchstaben können Kleinbuchstaben verwendet werden. In gleicher Weise sind die Abkürzungsmöglichkeiten für die übrigen Kommandos im Abschnitt 12.2.3. gekennzeichnet. Abkürzungen

sind nur in Kommandos zulässig; Schlüsselworte im BASIC-Programm müssen ausgeschrieben werden.
Das BYE-Kommando übergibt die Steuerung dem Betriebssystem. Die im BASIC-Arbeitsspeicher stehenden Informationen (Quellprogramm, abarbeitbares Zwischenprogramm, Fehlermeldungen usw.) werden auf eine Datei gerettet. Diese Datei wird auf dem aktuellen Gerät angelegt. Als Dateiname wird der Identifikationscode-verwendet. Der Dateityp ist DAT. Ist der Schalter /DE angegeben, geht der BASIC-Arbeitsspeicher verloren. Eventuell vorhandene Dateien werden gelöscht. BYE ohne den Schalter /DE ist zu verwenden, wenn:
-das ausführbare Zwischenprogramm erhalten werden soll,
- ein Programm übersetzt wurde, das anschließend von einem anderen Programm durch eine
  CHAIN-Anweisung gestartet werden soll,
- Informationen aus der Programmübersetzung (z. B. Fehlerermittlungen) erhalten werden sollen.
Benötigt man nur den Programmtext, kann dieser auch mit dem SAVE-Kommando gerettet werden, bevor das.Systern mit BYE verlassen wird.

Auf den Inhalt eines geretteten BASIC-Arbeitsspeichers kann man sich beim Start oder beim Restart des Systems beziehen. Ist beim Start des Systems auf dem aktuellen Gerät eine Datei mit dem angegebenen Identifikationscode und dem Dateityp. DAT vorhanden, wird aus dieser Datei der BASIC-Arbeitsspeicher aufgebaut. in diesem Fall erscheint vor der Kommandoanforderung "==>" noch die Ausschrift:

READ VS- FILE identifikationscode. DAT

Die weitere Arbeit kann dann bei dem Stand fortgesetzt werden, der beim zugehörigen Kommando
BYE /-DE bestand.
Mitunter müssen nacheinander mehrere BASIC-Programme bearbeitet und die dazugehörigen lauffähigen Zwischenprogramme aufbewahrt werden (z. B. bei der Korrektur von Programmen, die mit CHAIN verkettet sind). Dafür wird das RESTART-Kommando verwendet. Es hat das Format:

RESTART [/DE]

Beim RESTART-Kommando wird, wenn nicht der Schalter /DE angegeben ist, wie bei BYE /-DE, der BASIC-Arbeitsspeicher auf eine Datei gerettet. Anschließend wird wie beim Systemstart ein neuer Identifikationscode angefordert. Ist unter diesem neuen Identifikationscode der Abzug eines Arbeitsspeichers auf dem aktuellen Gerät verfügbar, wird daraus der BASIC-Arbeitsspeicher aufgebaut.

Kommandos

Nach der Aufforderung ==> erwartet das BASIC-System die Eingabe eines Kommandos, einer BASIC-Zeile oder einer Fortsetzungszeile zu einer unmittelbar vorher eingegebenen BASIC-Zelle. Eingegebene BASIC-Zellen werden automatisch sortiert. Die Kommandos werden durch den Kommandoprozessor entschlüsselt. Falls nötig, wird der das Kommando ausführende OVR-Teil in den Speicher geladen. Danach werden die Parameter des Kommandos ausgewertet und das Kommando ausgeführt. Zur Eingabe, Aufbereitung und Abarbeitung von BASIC-Programmen stehen eine Reihe weiterer Kommandos zur Verfügung.
Die Kommandos des Editormoduls haben unter Berücksichtigung der Abkürzungsmöglichkeit folgende Form:

LIST [liste-bereiche]
DELETE liste-bereiche
EXTRACT liste-bereiche
RENUMBER [z-bereich] AT z-num-1 [STEP schritt]

Das Löschen einzelner Zeilen kann auch durch Angabe der Zeilennummern, gefolgt vom EOL-Zeichen, erfolgen. Z. B. kann an Stelle des Kommandos

D 30,50 auch die Eingabefolge

30 (EOL)
50 (EOL)
verwendet werden.

Ist eine Zeile mit einer bestimmten Nummer bereits vorhanden, und es wird erneut eine Zeile mit dieser Nummer eingegeben, so wird die vorhandene Zeile überschrieben.

Das Neunumerieren des Programms mit dem RENUMBER-Kommando kann nur vorgenommen werden, wenn das Programm vorher übersetzt wurde und dabei keine Fehler auftraten. Diese Einschränkung sichert, daß bei der Neunumerierung auch alle Bezüge auf Zeilen korrekt erfaßt und geändert werden. Sollen bereits während der Programmeingabe größere Abschnitte zwischen vorhandene Zeilen eingefügt werden, ist eine entsprechende Numerierung mit größerer Schrittweite vorzusehen. Neben den Kommandos des Editormoduls sind die folgenden Kommandos realisiert:
PROTOCOL-Kommando

Das PROTOCOL-KOMMANDO stellt eine Erweiterung des, LIST-Kommandos dar. Es hat das Format:

PROTOCOL [dateispezifikation) [/liste-bereiche]

Die Dateispezifikation hat die Form:

[gerät:] [dateiname [.typ]]

Gerät , Dateiname und Dateityp sind entsprechend den Konventionen des Betriebssystems DCP anzugeben.
Mit dem PROTOCOL-Kommando wird das Protokoll der spezifizierten Programmabschnitte auf das angegebene Gerät oder die spezifizierte Datei ausgegeben. Neben dem Programmtext enthält das Protokoll an den entsprechenden Stellen Meldungen über lexikale, syntaktische und semantische Fehler bzw. Warnungen, die bei einer vorausgegangenen Übersetzung erkannt wurden. Die Protokollausgabe auf Dateien und Drucker erfolgt seitenweise. jede Seite hat eine Kopfzeile, die den Identifikationscode, das Datum und die Uhrzeit enthält. Fehlt die Dateispezifikation, erfolgt die Protokollausgabe über Terminal.
Unvollständige Dateispezifikationen werden wie folgt ergänzt:
- Dateiname und Typ fehlen:
Ist als Gerät CON: angegeben, erfolgt die Protokollausgabe über das Terminal, ist PRN: angegeben, über Drucker. Ansonsten ist als Gerät ein Diskettenlaufwerk, die Festplatte oder die virtuelle Speicherdiskette anzugeben, auf die das Protokoll als Datei ausgegeben wird. Als Dateiname wird der Identifikationscode verwendet. Für den Dateityp wird .LST ergänzt.
- Die Geräteangabe fehlt:
Als Gerät wird das aktuelle Gerät verwendet. Fehlt der Typ, wird .LST ergänzt.

Beispiele Der Identifikationscode sei CALC2, das aktuelle Gerät B:

P bewirkt die Ausgabe des vollständigen Protokolls über
Terminal.

PR A:	Das Protokoll wird als A: CALC2. LST abgespeichert.
PRO PRN: /600 TO LAST	Das Protokoll wird über Drucker ausgegeben. Es enthält die Zellen ab 600 bis Programmende.
PRO CALC3	Das Protokoll wird als B: CALC3. LST ausgegeben.

ERRORLIST-Kommando

Mit dem ERRORLIST-Kommando wird die Fehlerliste erzeugt.. Das Kommando hat das Format:

ERRORLIST [dateispezifikation]

Für jede fehlerhafte Zeile erscheinen der Text und anschließend die Meldungen über die bei dieser Zeile erkannten Fehler und Warnungen. Fehlt die Dateispezifikation, wird die Fehlerliste über Terminal ausgegeben. Ist die Dateispezifikation unvollständig, gelten die gleichen Regeln wie beim-PROTOCOL-Kommando.

SAVE-Kommando

SAVE [dateispezifikation) [/liste-bereiche]

Mit dem SAVE-Kommando wird der Text des aktuellen BASIC-Programms auf eine Datei oder ein Gerät ausgegeben. Das Kommando wird verwendet um den Text eines neu erstellten oder geänderten Programms zur späteren. Wiederverwendung zu sichern. Nach Änderung eines Programms sollte dieses stets auf Diskette oder auf Festplatte gesichert werden. Auf diese Weise werden Programmverluste bei Störungen vermieden. Fehlt die Angabe liste-bereiche, wird das ganze Programm gesichert. Ist unter einer angegebenen Dateispezifikation bereits eine Datei vorhanden, wird sie überschrieben.
Mit dem SAVE-Kommando erzeugte Dateien können mit dem Texteditor bearbeitet werden.
Unvollständige Dateispezifikationen werden wie folgt ergänzt:

- Dateiname und Typ fehlt:
  Falls als Gerät nicht CON: (Terminal) oder PRN: (Drucker) angegeben ist, wird als Dateiname der Identifikationscode und als Dateityp BAS ergänzt.
- Geräteangabe fehlt:
  Es wird das aktuelle Gerät verwendet. Fehlt der Dateityp, wird .BAS angenommen.
- Die gesamte Dateispezifikation fehlt:
  Der Programmtext wird unter dem Identifikationscode und dem Typ BAS auf Das aktuelle Gerät ausgegeben.

Beispiele Der Identifikationscode sei GRAPH, und das aktuelle Gerät ist C:

SAVE	bewirkt die Ausgabe als C: GRAPH.BAS
SAVE B:	GRAPH.BAS wird auf B: ausgegeben.
SAVE PRN:	Der Programmtext wird ohne Fehlermeldungen und ohne Seiteneinteilung gedruckt.
SA A:LIB.EXE/ 10000 TO LAST	Der Programmtext ab Zeile 10000 bis zum Programmende wird unter dem Namen LIB.EXE auf A: ausgegeben.

OLD-Kommando

Mit dem OLD-Kommando
OLD [dateispezifikation]

werden BASIC-Programme von Dateien eingelesen. Die Dateien können mit dem SAVE-Kommando oder mit dem Texteditor erstellt worden sein. Vor dem Einlesen eines neuen Programms werden ein eventuell im BASIC-Arbeitsspeicher stehendes BASIC-Programm und die dazu gehörigen Informationen gelöscht. Fehlt die Dateispezifikation vollständig, oder ist sie unvollständig, werden die gleichen Annahmen wie beim SAVE-Kommando getroffen.

Beispiele Der Identifikationscode sei PLOT und das aktuelle Gerät A:

```
OLD              liest das Programm von A: PLOT.BAS ein.
0 B: F2          Das neue aktuelle Programm wird von B: F2.BAS gelesen.
```

ADD-Kommando

Das Kommando
ADD [dateispezifikation]

wirkt wie das OLD-Kommando, jedoch wird ein im BASIC-Arbeitsspeicher bereits vorhandenes Programm nicht gelöscht. Das in der Dateispezifikation angegebene Programm wird zum vorhandenen Programm hinzugefügt.

Enthalten die spezifizierte Datei und das vorhandene aktuelle Programm Zeilen gleicher Nummer, werden die entsprechenden Zeilen des aktuellen Programms durch die eingelesenen Zeilen überschrieben. ADD mischt das aktuelle Programm mit den eingelesenen Zeilen. Das ADD-Kommando kann verwendet werden, um mehrfach verwendete externe Funktionen oder Prozeduren an verschiedene Hauptprogramme anzufügen. Dazu müssen die Zeilennummern geeignet gewählt werden.

Beispiel
```
OLD MAIN
ADD F:SCREEN.BIB
```

An das Hauptprogramm MAIN wird der Programmteil SCREEN.BIB angefügt. Werden aus SCREEN.BIB nur einige Abschnitte (z. B. die Zeilen *7300* bis *7800* und 9600 bis 9900) benötigt, können folgende Kommandos benutzt werden:

```
0 F:SCREEN.BIB
EX 7300 TO 7800,9600 TO 9900
AD MAIN
```

RUN-Kommando
Das RUN-Kommando veranlaßt die Abarbeitung des aktuellen Programms:

RUN

Existiert im BASIC-Arbeitsspeicher ein gültiges abarbeitbares Zwischenprogramm, wird dieses in den Hauptspeicher geladen und gestartet. War das aktuelle BASIC-Programm vorher neu eingegeben oder geändert worden, wird es zunächst übersetzt. Dabei wird ein abarbeitbares Zwischenprogramm erzeugt. Wenn bei der Übersetzung Fehler erkannt werden, kann das erzeugte Zwischenprogramm nicht ausgeführt werden. Die Fehlermeldungen können mit PROTOCOL oder ERRORLIST ausgegeben werden. Erst nach Beseitigung der Fehler ist ein erneutes RUN-Kommando sinnvoll.

TRANSLATE-Kommando
Das TRANSLATE-Kommando

TRANSLATE

veranlaßt die Übersetzung des aktuellen Programms, ohne daß das Programm anschließend geladen und gestartet wird. Ist das Programm fehlerfrei, entsteht ein abarbeitbares Zwischenprogramm.

Beispiel
Das Programm MENUE startet über CHAIN-Anweisungen die Programme SUCHE und AUFN. Bevor MENUE abgearbeitet werden kann, müssen für AUFN und SUCHE abarbeitbare Zwischenprogramme erstellt und mit RESTART abgelegt werden. Wenn alle Programme korrekt sind, ergibt sich folgender Dialog:

```
/>BAS
Robotron Standard-BASIC, Version xx.xx DCP, 1988

IDENTIFICATION: SUCHE
=>   OLD
=>   T
PASS 1
PASS 2
=>   RES
IDENTIFICATION: AUFN
=>   OLD
=>   T
PASS 1
PASS 2
=>   RES
IDENTIFICATION: MENUE
=>OLD
=>RUN
PASS 1
PASS 2
```

... es folgt die Abarbeitung von MENUE.

START-Kommando

START

wird verwendet, wenn ein Programm mehrfach nacheinander abgearbeitet werden soll. Es wird sofort das im Hauptspeicher stehende Zwischenprogramm abgearbeitet. Das La den des arbeitsfähigen Zwischenprogramms aus dem BASIC-Arbeitsspeicher in den Hauptspeicher wird eingespart. Ist im Hauptspeicher kein arbeitsfähiges Zwischenprogramm vorhanden, wird das RUN-Kommando ausgeführt.

Fehlermitteilungen

Auf Fehler bei der Eingabe, der Korrektur und der Abarbeitung des Programms und auf die Verletzung der syntaktischen und semantischen Regeln durch den Programmtext reagiert das System mit Fehlermeldungen. An dieser Stelle können nur die wichtigsten Gruppen von Meldungen erwähnt werden. Eine detaillierte Darstellung enthält die *Anteitung für den Bediener* zum BASIC-System.

Fehler bei der Arbeit mit dem Programmtext
Solche Fehler werden gemeldet, wenn eine Eingabezeile weder mit einer gültigen Zeilennummer noch mit Ampersand (Fortsetzungszeile) oder mit einem Kommandoschlüsselwort beginnt, wenn unzulässige Parameter im Kommando stehen oder EA-Fehler bei der Ausführung von SAVE-, OLD-, ADD oder BYE/-DE-Kommandos auftreten.

Beispiele

```
60000 LET A= -2
-ILLEGAL LINE NUMBER
     ENTER THE CORRECT LINE NUMBER
          (5 DIGITS AT MOST; >0; < =50000)
:6000
```

Die, Zeilennummer ist größer als 50000. Es wird eine neue Zeilennummer angefordert, unter der die Zeile eingefügt wird.

```
OLD B:ITER
-FAILED TO OPEN OLD FILE
SYSTEM ERROR CODE:      02
```

Beim Einlesen des Programms ITER.BAS von B: tritt ein EA-Fehler auf. Die Ursache ist anhand des Fehlercodes und der Betriebssystemdokumentation zu, ermitteln.

Fehler bei der Übersetzung von Programmen

Bei der Übersetzung werden erkannte Fehler und Warnungen am Ende der Übersetzung mit der Ausschrift

ERRORS xx
WARNINGS xx
signalisiert.

Die Programmzeilen, bei denen Fehler erkannt wurden, können mit dem ERRORLIST bzw. dem PROTOCOL-Kommando ermittelt werden. Enthält das Programm nur Warnungen, ist PROTOCOL zu verwenden. Nach jeder Zeile erscheint dabei jeweils eine Unterstreichungszelle, die angibt, bei weichem Zeichen der betroffenen Zeile der Fehler bemerkt wurde. Anschließend folgt die Fehlerausschrift. Bei Warnungen besteht die Unterstreichungszeile aus Pluszeichen, bei Fehlern aus Sternzeichen.

Beispiel

```
30 LET AX2$ =" Art. -Nr.
++++++++++++++++++++
QUOTE IS COMPLETED
```

Das fehlende Anführungszeichen wird in diesem Fall automatisch ergänzt. Meldungen syntaktischer Fehler beginnen stets mit dem Wort MISSING. Danach werden durch Schrägstriche diejenigen Zeichen und Symbole angeführt, von denen eines an der Abbruchstelle korrekterweise hätte stehen müssen.

Beispiel

```
30 AX2$ ="ART. - NR."
**********************
MISSING:
\END\LET\GOTO\GOSUB\GO\PRINT\INPUT\ZINE\OPEN\CLOSE\SET\ASK
```

Die Anweisung muß mit einem Schlüsselwort beginnen. An dieser Stelle ist es LET. Es werden nur soviele Fortsetzungsmöglichkeiten angegeben, wie auf einer Zeile untergebracht werden können.
Bei syntaktischen (MISSING-)Fehlern und lexikalen Fehlern (zu langen Identifikatoren, unzulässigen Zeichen in Stringkonstanten, Überlauf bei der Konvertierung numerischer Konstanten) bezeichnet die Unterstreichungszeile genau die Stelle, die den Fehler verursacht hat. Bei den übrigen,Fehlern bezeichnet sie nur die Stelle, an der der Fehler bemerkt wurde. Die Fehlerursache liegt dann entweder im unterstrichenen Teil der gleichen Zeile oder in vorhergehenden Zeiten (z. B. bei Fehlern, die erst am Ende der Programmeinheit erkannt werden können).

Beispiel

```
2300 EXTERNAL FUNCTION SEQ (P1 ( , ),     P2)
       :
     2390 END FUNCTION
     *********************
                  FUNCTION RESULT HAS LEFT BEEN UNDEFINED
```

In der externen Funktion fehlt die Zuweisung des Funktionswertes an den Funktionsnamen. Fehler in Deklarationen und Parameterlisten können Folgefehler verursachen.

Fehler bei der Abarbeitung von Programmen
Bei der Abarbeitung eines Programms werden triviale und nicht triviale Ausnahmen über das Terminal gemeldet. Die Meldung enthält die Nummer der Zeile, an der die Ausnahme auftrat, und den Ausnahmecode. Tritt eine Ausnahme bei Terminaleingabe auf, ist eine Korrektur der verursachenden Eingabezeile möglich.

A. Standardisierter Zeichenvorrat von BASIC

Der standardisierte BASIC-Zeichenvorrat wird durch den ANSI-Standard ANSI 3.4-1977 (ASCII), bzw. den ISO-Standard ISO 646 (7-bit-Code) definiert.

1.Die Buchstaben haben die Ordnungspositionen 65-90 und 97-122, die Ziffern die Positionen 48-57. Die in einem Programmtext verwendbaren Sonderzeichen belegen die Positionen 32-47, 58-63, 94 und 95.

2.Die Sonderzeichen auf den Positionen: 64, 91-93, 96 und 123-126 werden für eine nationale Verwendung reserviert und sind als solche in einem BASIC-Programmtext nicht zulässig. Sie stellen in der internationalen Referenz-Version des Zeichenvorrats folgende Zeichen dar.

@ [\] ´ {|}

Die Norm DIN 66 003 (deutsche Version des Zeichenvorrats) definiert an diesen Stellen die Zeichen

§ Ä Ö Ü ä ö ü ß

3.Die Steuerzeichen nehmen die Positionen 0-32 und 127 ein. Das Leerzeichen -Position 32 - ist ein nicht druckbares Schriftzeichen.

4.Eine konkrete Prozessorrealisierung kann weitere Zeichen mit Ordnungszahlen größer als 127 definieren. Solche Zeichen dürfen, wie die im Punkt 2 genannten Zeichen, angewendet werden.

Ordnungsposition	Kurzzeichen (bei Steuerzeichen) oder Schriftzeichen	Benennung
0	NUL	Nil
1	SOH	Anfang des Kopfes
2	STX	Anfang des Textes
3	ETX	Ende des Textes
4	EOT	Ende des Übertragung
5	ENQ	Stationsaufforderung
6	ACK	Positive Rückmeldung
7	BEL	Klingel
8	BS	Rückwärtsschritt
9	HT	Horizontal-Tabulator
10	LF	Zellenvorschub
11	VT	Vertikal-Tabulator
12	FF	Formular-Vorschub
13	CR	Wagenrücklauf
14	so	Dauerumschaltung
15	SI	Rückschaltung
16	DLE	Datenübertragungsumschaltung
17	DC1	
18	DC2	Gerätesteuerzeichen
19	DC3	
20	DC4	
21	NAK	Negative Rückmeldung
22	SYN	Synchronisierung
23	ETB	Ende des Datenblockes
24	CAN	Ungültig
25	EM	Ende der Aufzeichnung
26	SUB	Substitutionszeichen
27	ESC	Code-Umschaltung
28		FS

29	GS	
	Informationstrennzeichen	
30	RS	
31	US	
32	SP	Leerzeichen,
	Zwischenraum	
33	!	Ausrufungszeichen
34	„	Anführungszeichen
35	#	Nummernzeichen
36	$	Dollar
37	%	Prozent
38	&	Kommerzielles Und
39	´	Apostroph
40	(	runde Klammer auf
41	)	runde Klammer zu
42	*	Stern
43	+	plus
44	,	Komma
45	-	minus, Bindestrich
46	.	Punkt
47	/	Schrägstrich

Anlage A

ListederKommandos,AnweisungenundFunktioneninalphabe-
tischerReihenfolge

Syntax	Wirkungsweise
ABS(x)	Uebergibt den absoluten Wert des Ausdrucks x.
ASC(x$)	Uebergibt den ASCII-Code fuer das erste Zeichen der Zeichenkette x$
ATN(x)	Uebergibt den Arcustangens von x.
AUTO [Nummer] [,[Schrittweite]]	Erzeugt automatisch Zeilennummern
BEEP	Lautsprecher erzeugt einen Ton.
BLOAD Dateiangabe [,Relativzeiger]	Laedt eine Datei mit einem Hauptspeicherabbild (z.B. Programme in Maschinensprache) in den Hauptspeicher.
BSAVE Dateiangabe ,Relativzeiger,Laenge	Speichert Teile des Hauptspeichers in der angegebenen Datei.
CALL Numerische Variable [(Variable[, Variable]...)]	Ruft ein Unterprogramm in Maschinensprache auf.
CDBL(x)	Wandelt x in eine Zahl doppelter Genauigkeit um.
CHAIN [MERGE] Dateiangabe [,[Zeile][,ALL] [,DELETE Bereich]]	Ruft ein anderes Programm auf, uebertraegt die Steuerung an dieses und uebergibt Variable aus dem laufenden Programm.

CHDIR Pfad	Ermoeglicht das Aendern des laufenden Verzeichnisses.
CHR$(n)	Uebergibt das Zeichen mit dem ASCII-Code n.
CINT(x)	Wandelt x in eine ganze Zahl um durch Runden.
CIRCLE (x,y),r[,Farbe [,Start,End[,Bild]]]	Zeichnen einer Ellipse auf dem Bildschirm mit dem Mittelpunkt (x,y) und dem Radius r.
CLEAR [,[n][,m]]	Loescht die Variablen, wahlfrei kann die Groesse des Arbeitsspeichers und des Stapelbereiches definiert werden.
CLOSE [[#]Dateinummer,...]	Schliesst die Ein-/Ausgabe fuer eine Einheit oder Datei ab.
CLS	Loescht den Bildschirm.
COLOR [Vordergrund] [,[Hintergrund] [,Bildschirmrand]]	Setzt im Textmodus die Farben fuer den Vordergrund, Hintergrund und den Bildschirmrand.
COLOR [Hintergrund] [,[Palette]]	Setzt im Grafikmodus die Farbe fuer den Hintergrund und waehlt eine von zwei Paletten aus.
COM(n) ON/OFF/STOP	Aktiviert oder inaktiviert die Unterbrechung fuer Datenfernverarbeitungsaktivitaeten.
COMMON Variable [,Variable]...	Uebergibt Variable an das durch CHAIN aufgerufene Programm.
CONT	Nimmt die Programmausfuehrung nach einer Unterbrechung wieder auf.
COS(x)	Errechnet und uebergibt den Cosinuswert von x.
CSNG(x)	Wandelt x in eine Zahl einfacher Genauigkeit um.
CSRLIN	Uebergibt die vertikale Koordinate des Kursors.
CVI(2-Byte-Zeichenkette) CVS(4-Byte-Zeichenkette) CVD(8-Byte-Zeichenkette)	Wandelt die Zeichenkette in einen numerischen Wert um.
DATA Konstante [,Konstante]...	Speichert numerische und Zeichenkettenkonstanten, die mit der Anweisung READ im Programm gelesen werden koennen.
DATE$	Liest das Datum.
DATE$=x$	Setzt das Datum.

DEF FNName[(Arg[,Arg]...)] = Ausdruck	Definiert eine Funktion.
DEF SEG [=Segment]	Definiert das laufende Segment des Hauptspeichers.
DEFTyp Buchstabe[-Buchstabe] [,Buchstabe [-Buchstabe]]...	Definiert Variablentypen als ganzzahlig, einfache Genauigkeit, doppelte Genauigkeit oder Zeichenketten. Wobei Typ sein kann: INT, SNG, DBL, STR.
DEF USR(n)=Relativzeiger	Definiert die Startadresse eines Unterprogramms in Maschinensprache.
DELETE [Zeile1][-Zeile2] DELETE [Zeile1-]	Loescht den angegebenen Programmzeilenbereich.
DIM Variable(Indizes) [,Variable(Indizes)]...	Definiert die maximalen Werte fuer die Indizes von Bereichsvariablen und ordnet den erforderlichen Speicherplatz zu.
DRAW Zeichenkette	Zeichnet ein Objekt, das durch eine Zeichenkette angegeben ist.
EDIT Zeile	Zeigt eine Programmzeile zur Aenderung an.
END	Beendet die Programmausfuehrung, schliesst alle Dateien ab und kehrt zur Befehlsebene zurueck.
ENVIRON Parameter=Zeichenkette	Aendert Parameter in der BASIC-Umgebungstabelle
ENVIRON$ (Parameter$) ENVIRON$ (n)	Zeigt eine angegebene Zeichenkette der BASIC-Umgebungstabelle an.
EOF (Dateinummer)	Zeigt eine Dateiendebedingung an.
ERASE Feldname [,Feldname]...	Loescht Felder aus einem Programm.
ERDEV	Variable, die den Unterbrechungsfehlercode INT24 fuer einen Einheitenfehler enthaelt.
ERDEV$	Variable, die den Namen der Einheit enthaelt, die den Fehler verursachte.
ERL	Uebergibt die Zeilennummer, in der der letzte Fehler aufgetreten ist.
ERR	Uebergibt den Fehlercode fuer den letzten Fehler.
ERROR n	Simuliert das Auftreten eines BASIC-Fehlers oder ermoeglicht die Definition eigener Fehlercodes.
EXP(x)	Errechnet die Exponentialfunktion.

FIELD [#]Dateinummer, Laenge AS Zeichenkettenvariable [,Laenge AS ZK-Variable]...	Legt Platz fuer Variable in einem Puffer fuer Dateien mit wahlfreiem Zugriff an.
FILES [Dateiangabe]	Zeigt die Dateinamen auf einer Diskette an.
FIX(x)	Schneidet x auf eine ganze Zahl ab.
FOR Variable=x TO y [STEP z]	Fuehrt eine Folge von Anweisungen in einer Schleife aus. Die Anweisung NEXT schliesst die Schleife ab.
FRE(x) FRE(x$)	Uebergibt die Anzahl der Byte im Hauptspeicher, die nicht von BASIC benutzt werden.
GET [#]Dateinummer [,Nummer]	Liest einen Satz aus einer Datei mit wahlfreiem Zugriff in einen Puffer fuer wahlfreien Zugriff.
GET (x1,y1)-(x2,y2), Bereich	Liest Punkte aus einem Bereich des Bildschirms.
GOSUB Zeile	Verzweigen in ein Unterprogramm. Mit Hilfe der Anweisung RETURN erfolgt die Rueckkehr aus dem Unterprogramm.
GOTO Zeile	Unbedingte Verzweigung aus dem normalen Programmablauf zu einer angegebenen Zeilennummer.
HEX$(n)	Wandelt den dezimalen Wert n in eine hexadezimale Zeichenkette um .
IF Ausdruck[,] THEN Angabe [ELSE Angabe] IF Ausdruck[,] GOTO Zeile [[,]ELSE Angabe]	Aendert den Programmablauf, abhaengig vom Ergebnis eines Ausdrucks.
INKEY$	Liest ein Zeichen von der Tastatur.
INP(n)	Uebergibt ein vom Tor n gelesenes Byte.
INPUT[;]["Anfrage";] Variable[,Variable]...	Liest Daten von der Tastatur.
INPUT #Dateinummer,Variable [,Variable]...	Liest Daten aus einer sequentiellen Einheit oder Datei und weist sie Programmvariablen zu.
INPUT$(n[,[#]Dateinummer])	Uebergibt eine Zeichenkette von n Zeichen, die von der Tastatur oder von einer Datei mit der Nummer Dateinummer gelesen wurde.

INSTR([n,]x$,y$)	Sucht das erste Auftreten der Zeichenkette y$ in x$ und uebergibt die Position, an der die Zeichenkette gefunden wurde. n setzt die Position fest, an der mit der Suche begonnen werden soll.
INT(x)	Uebergibt die groesste ganze Zahl, die kleiner oder gleich x ist.
IOCTL [#]Dateinummer, Zeichenkette	Sendet eine Steuerdatenzeichenkette an einen Einheitentreiber.
IOCTL$([#]Dateinummer)	Liest eine Steuerdatenzeichenkette von einem eroeffneten Einheitentreiber.
KEY ON/OFF/LIST	Zeigt den Inhalt der Funktionstasten an oder loescht die Anzeige.
KEY n,x$	Ordnet den Wert x$ der angegebenen Funktionstaste zu.
KEY n, CHR$(Tastenkennzeichen) + CHR$ (Suchcode)	Definiert zusaetzliche Funktionstasten.
KEY(n) ON/OFF/STOP	Aktiviert oder inaktiviert die Unterbrechung fuer eine angegebene Taste in einem BASIC-Programm.
KILL Dateiangabe	Loescht eine Datei von der Diskette.
LEFT$(x$,n)	Uebergibt die am weitesten links liegenden n Zeichen aus x$.
LEN(x$)	Uebergibt die Anzahl der Zeichen in x$.
LET Variable=Ausdruck	Ordnet den Wert eines Ausdrucks einer Variablen zu.
LINE [(x1,y1)]-(x2,y2) [,[Farbe][,B[F]] [,Stil]]	Zeichnet eine Linie oder ein Viereck auf dem Bildschirm.
LINE INPUT[;]["Anfrage";] Zeichenkettenvariable	Liest eine Zeile von der Tastatur in eine Zeichenkettenvariable.
LINE INPUT # Dateinummer, Zeichenkettenvariable	Liest eine Zeile von einer sequentiellen Datei in eine Zeichenkettenvariable.
LIST [Zeile1][-[Zeile2]] [,Dateiangabe]	Listet das im Hauptspeicher stehende Programm auf dem Bildschirm oder auf einer anderen angegebenen Einheit auf.
LLIST [Zeile1][-[Zeile2]]	Listet das gesamte oder einen Teil des Programms auf dem Drukker aus.

LOAD Dateiangabe[,R]	Laedt ein Programm von einer angegebenen Einheit in den Hauptspeicher und fuehrt es wahlweise aus.
LOC(Dateinummer)	Uebergibt die aktuelle Position in der Datei.
LOCATE[Zeile][,[Spalte][,[Anzeige][,[Start][,Stopp]]]]	Setzt den Kursor auf dem aktiven Bildschirm. Wahlfreie Parameter schalten das Blinken des Kursors an und aus und definieren die Groesse des blinkenden Kursors.
LOF(Dateinummer)	Uebergibt die Anzahl Bytes, die einer Datei zugeordnet sind (Laenge der Datei).
LOG(x)	Uebergibt den natuerlichen Logarithmus von x.
LPOS(n)	Uebergibt die laufende Position des Druckkopfs im Druckpuffer.
LPRINT [Liste von Ausdruecken[;]]	Gibt Daten auf dem Drucker aus.
LPRINT USING v$; Liste von Ausdruecken[;]	Gibt Daten auf dem Drucker aus mit Hilfe des durch v$ angegebenen Formates.
LSET Zeichenkettenvariable=x$	Uebergibt Daten in einen Dateipuffer fuer Direktzugriff (linksbuendig).
MERGE Dateiangabe	Mischt Zeilen aus einer ASCII-Programmdatei in ein Programm, das sich im Hauptspeicher befindet.
MID$(x$,n[,m])	Uebergibt m Zeichen aus x$, beginnend an der Position n.
MKDIR Pfad	Erstellt ein Verzeichnis auf einer angegebenen Diskette.
MKI$(ganzzahliger Ausdruck) MKS$(Ausdruck einf. Genauigkeit) MKD$(Ausdruck dopp. Genauigkeit)	Wandelt numerische Werte in Zeichenkettenwerte um.
NAME Dateiangabe AS Dateiname	Aendert den Namen einer Diskettendatei.
NEW	Loescht das im Hauptspeicher befindliche Programm und alle Variablen.
NEXT[Variable[,Variable]...]	Schliesst eine FOR...NEXT-Schleife ab.
OCT$(n)	Uebergibt eine Zeichenkette, die den oktalen Wert eines dezimalen Arguments darstellt.

ON COM(n) GOSUB Zeile	Setzt fuer BASIC eine Zeilennummer, zu der verzweigt wird, sobald eine Inf. im Datenfernverarbeitungspuffer steht.
ON ERROR GOTO Zeile	Aktiviert die Fehlerunterbrechung und gibt die erste Zeile des Fehlerbehandlungsprogramms an.
ON n GOTO Zeile[,Zeile]... ON n GOSUB Zeile[,Zeile]...	Verzweigt zu einer von mehreren angegebenen Zeilennummern, ab haengig vom Wert des Ausdruckes.
ON KEY(n) GOSUB Zeile	Aktiviert die Unterbrechung, falls eine angegebene Funktions- oder Kursortaste betaetigt wurde und gibt die erste Zeile der Unterbrechungsroutine an.
ON PLAY(n) GOSUB Zeile	Erlaubt fortlaufende Musik waehrend der Programmausfuehrung.
ON TIMER(n) GOSUB Zeile	Uebergibt nach Ende einer angegebenen Zeitspanne die Steuerung an eine angegebene Zeilennummer.
OPEN Dateiangabe [FOR Modus] AS [#]Dateinummer [LEN=Satzlaenge] OPEN Modus2,[#]Dateinummer, Dateiangabe[,Satzlaenge]	Erlaubt die Ein-/Ausgabe zu und von einer Datei oder Einheit.
OPEN "COMn:[Geschwindigkeit] [,Paritaet][,Daten] [,Stopp][,RS][,CS[n]] [,DS[n]][,CD[n]][,LF] [,PE]" AS[#]Dateinummer [LEN=Nummer]	Eroeffnet eine Datei fuer Datenfernverarbeitung.
OPTION BASE n	Definiert den kleinsten Wert fuer Feldindizes.
OUT n,m	Sendet ein Byte zu einem Maschinenausgabetor.
PAINT (x,y)[[,Farbe][,Rand] [,Hintergrund]]	Fuellt einen Bildschirmbereich mit einer ausgewaehlten Farbe.
PEEK(n)	Uebergibt ein gelesenes Byte von der angegebenen Hauptspeicherposition.
PLAY Zeichenkette	Spielt Musik mit Hilfe einer Zeichenkette.
PLAY(n)	Uebergibt die Anzahl der Noten, die sich gerade im Puffer fuer Hintergrundmusik befinden.
PMAP(x,n)	Rechnet physische Koordinaten in Weltkoordinaten und Weltkoordinaten in physische Koordinaten um.
POINT(x,y) POINT(n)	Uebergibt die Farbe eines angegebenen Punktes auf dem Bildschirm oder die laufende grafische Koordinate oder den Wert der

	laufenden x- oder y-Koordinaten.
POKE n,m	Schreibt ein Byte in eine Haupt speicherstelle.
POS(n)	Uebergibt die Spaltenposition, in der sich der Kursor gerade befin det.
PRINT [Liste der Ausdruecke] [;] ? [Liste der Ausdruecke][;]	Zeigt Daten auf dem Bildschirm an.
PRINT USING v$; Liste der Ausdruecke[;]	Gibt Daten mit Hilfe eines For mates aus.
PRINT #Dateinummer, [USING v$;] Liste der Ausdruecke	Schreibt Daten sequentiell in ei ne Datei.
PSET(x,y)[,Farbe] PRESET(x,y)[,Farbe]	Zeichnet einen Punkt an eine an gegebene Stelle auf dem Bild schirm.
PUT [#] Dateinummer [,Nummer]	Schreibt einen Satz aus dem Puf fer fuer Direktzugriff in eine Direktzugriffsdatei.
PUT (x,y), Bereich[,Aktion]	Faerbt einen angegebenen Bereich des Bildschirms.
RANDOMIZE[n] RANDOMIZE TIMER	Uebergibt einen neuen Anfangswert fuer den Zufallszahlengenerator.
READ Variable[,Variable]...	Liest Werte aus der Anweisung DATA und ordnet sie Variablen zu.
REM Bemerkung	Einfuegen erklaerender Bemer kungen in ein Programm.
RENUM [neue Nummer] [,[alte Nummer] [,Schrittweite]]	Neunumerierung von Programmzei len.
RESET	Schliesst alle Diskettendateien ab und loescht den Systempuffer.
RESTORE [Zeile]	Erlaubt es, DATA-Anweisungen ab einer angegebenen Zeile wieder zu lesen.
RESUME [0] RESUME NEXT RESUME Zeile	Faehrt mit der Programmausfueh rung fort, nachdem eine Fehlerbe handlungsroutine ausgefuehrt wur de.
RETURN [Zeile]	Verzweigt aus einem Unterprogramm zurueck.
RIGHT$(x$,n)	Uebergibt die rechten n Zeichen der Zeichenkette x$.
RMDIR Pfad	Loescht das Verzeichnis von der angegebenen Diskette.
RND[(x)]	Uebergibt eine Zufallszahl zwi schen 0 und 1.

RSET Zeichenkettenvariable =x$	Uebergibt Daten in einen Dateipuffer fuer Direktzugriff (rechtsbuendig).
RUN [Zeile] RUN Dateiangabe[,R]	Beginnt die Programmausfuehrung.
SAVE Dateiangabe[,A] SAVE Dateiangabe[,P]	Speichert eine BASIC-Programmda tei auf Diskette.
SCREEN (Zeile,Spalte[,z])	Uebergibt den ASCII-Code (0 bis 255) fuer ein Zeichen auf dem aktiven Bildschirm an der angege benen Zeile und Spalte.
SCREEN [Modus][[Aendern] [,[,Aseite] [,Vseite]]]	Setzt die Bildschirmattribute, die von den nachfolgenden Anwei- sungen benutzt werden sollen.
SGN(x)	Uebergibt das Vorzeichen von x.
SHELL [Befehlszeichenkette]	Laedt eine andere Programmdatei und fuehrt das Programm aus.
SIN(x)	Errechnet und uebergibt den Sinuswert fuer x (x im Bogen mass).
SOUND Freq., Dauer	Erzeugt Toene ueber den Laut sprecher.
SPACE$(n)	Uebergibt eine Zeichenkette, die aus n Leerstellen besteht.
SPC(n)	Druckt in einer PRINT- oder LPRINT-Anweisung n Leerstellen.
SQR(x)	Uebergibt die Quadratwurzel von x.
STOP	Beendet die Programmausfuehrung und kehrt zur Befehlsebene zu rueck.
STR$(x)	Uebergibt die Zeichenkettendar stellung des Wertes x.
STRING$(n,m) STRING$(n,x$)	Uebergibt eine Zeichenkette der Laenge n, deren Zeichen alle aus dem ASCII-Code m oder dem ersten Zeichen von x$ bestehen.
SWAP Variable1,Variable2	Tauscht die Werte zweier Variab len aus.
SYSTEM	Verlaesst BASIC und kehrt zu DCP zurueck.
TAB(n)	Setzt in Verbindung mit der An weisung PRINT an der Position n einen Tabulator.
TAN(x)	Uebergibt den Tangens von x.
TIME$	Uebergibt die laufende Zeit.

TIME$=x$	Setzt die laufende Zeit.
TIMER	Uebergibt einen Wert einfacher Genauigkeit, der die Sekunden darstellt, die seit Mitternacht oder einem Warmstart des Systems vergangen sind.
TRON/TROFF	Zeigt eine Ablaufverfolgung der Programmanweisungen bei der Aus fuehrung des Programms an.
USR[n](arg)	Ruft das angegebene Unterprogramm in Maschinensprache mit dem Argu ment arg auf.
VAL(x$)	Uebergibt den numerischen Wert der Zeichenkette x$.
VARPTR(Variable)	Uebergibt den Relativzeiger zu dem aktuellen Speichersegment der Variablen.
VARPTR$(Variable)	Uebergibt die Zeichenform der Adresse einer Variablen im Haupt speicher.
VIEW[[SCREEN][(x1,y1) -(x2,y2)[,[Farbe] [,Rand]]]]	Definiert einen rechteckigen Aus-schnitt des Bildschirms.
WAIT Tor,n[,m]	Verlaesst die Programmausfuehrung zur Beobachtung des Status eines Maschineneingabetores.
WEND	Schliesst die Anweisung WHILE ab.
WHILE Ausdruck	Fuehrt eine Serie von Anweisungen in einer Schleife so lange aus, wie eine angegebene Bedingung richtig ist.
WIDTH Groesse WIDTH Einheit,Groesse WIDTH #Dateinummer, Groesse	Legt die Anzahl der Zeichen einer Ausgabezeile fest. Nachdem die angegebene Anzahl von Zeichen ausgegeben wurde, fuegt BASIC eine Zeilenschaltung hinzu.
WINDOW [[SCREEN] (x1,y1)-(x2,y2)]	Erlaubt die Neudefinition der Bildschirmkoordinaten.
WRITE [Liste der Ausdruecke]	Gibt Daten auf dem Bildschirm aus.
WRITE #Dateinummer, Liste der Ausdruecke	Schreibt Daten in eine sequen tielle Datei.

Anlage B

Reservierte Woerter

Einige Woerter bei BASIC haben eine bestimmte Bedeutung. Diese Woerter nennt man Reservierte Woerter. Reservierte Woerter beinhalten alle BASIC-Kommandos, -Anweisungen, -Funktionsnamen und -Operatornamen. Reservierte Woerter duerfen nicht als Variablennamen benutzt werden. Reservierte Woerter muessen immer von Daten oder anderen Teilen der BASIC-Anweisung durch Leerstellen oder andere Sonderzeichen, die in der Syntax erlaubt sind, getrennt werden. Das bedeutet, dass Reservierte Woerter immer getrennt stehen muessen, so dass sie von BASIC erkannt werden koennen.

Die folgende Liste zeigt alle reservierten Woerter:

ABS	ENVIRON	LOC	Space$
AND	ENVIRON$	LOCATE	
ASC	EOF	LOF	
ATN	EQV	LOG	
AUTO	ERASE	LPOS	
BEEP	ERDEV	LPRINT	
BLOAD	ERDEV$	LSET	
BSAVE	ERL	MERGE	
CALL	ERR	MID$	
CDBL	ERROR	MKDIR	
CHAIN	EXP	MKD$	
CHDIR	FIELD	MKI$	
CHR$	FILES	MKS$	
CINT	FIX	MOD	
CIRCLE	FNxxxxxxxx	NAME	
CLEAR	FOR	NEW	
CLOSE	FRE	NEXT	
CLS	GET	NOT	
COLOR	GOSUB	OCT$	
COM	GOTO	OFF	
COMMON	HEX$	ON	
CONT	IF	OPEN	
COS	IMP	OPTION	
CSNG	INKEY$	OR	
CSRLIN	INP	OUT	
CVD	INPUT	PAINT	
CVI	INPUT$	PEEK	
CVS	INPUT#	PLAY	
DATA	INSTR	PMAP	
DATE$	INT	POINT	
DEF	IOCTL	POKE	
DEFDBL	IOCTL$	POS	
DEFINT	KEY	PRESET	
DEFSNG	KILL	PRINT	
DEFSTR	LEFT$	PRINT#	
DELETE	LEN	PSET	
DIM	LET	PUT	
DRAW	LINE	RANDOMIZE	
EDIT	LIST	READ	
ELSE	LLIST	REM	
END	LOAD	RENUM	
RESET	SPC	TRON	
RESTORE	SQR	USING	
RESUME	STEP	USR	
RETURN	STOP	VAL	
RIGHT$	STR$	VARPTR	
RMDIR	STRING$	VARPTR$	
RND	SWAP	VIEW	
RSET	SYSTEM	WAIT	
RUN	TAB	WEND	
SAVE	TAN	WHILE	
SCREEN	THEN	WIDTH	
SGN	TIME$	WINDOW	

SHELL	TIMER	WRITE
SIN	TO	WRITE#
SOUND	TROFF	XOR

ANLAGE C

Zusammenstellung der Fehlercodes und der Fehlermeldungen

Falls BASIC einen Fehler entdeckt, der das Programm stoppt, wird eine Fehlernachricht angezeigt. Es ist moeglich, mit der Anweisung ON ERROR und den Variablen ERR und ERL Fehler in einem BASIC-Programm zu lesen und zu testen. (Eine vollstaendige Erklaerung fuer ON ERROR, ERR, ERL, ERDEV und ERDEV$ steht unter der jeweiligen Anweisung oder Funktion in diesem Handbuch.

In diesem Anhang sind alle BASIC-Fehlernachrichten mit ihren zugehoerigen Fehlernummern aufgefuehrt:

Nummer	Nachricht
1	NEXT without FOR (NEXT ohne FOR). Der Anweisung NEXT fehlt die zugehoerige Anweisung FOR. Es kann sein, dass eine Variable in der Anweisung NEXT nicht mit einer zuvor ausgefuehrten und ungleichen Anweisung FOR-Variablen uebereinstimmt. Das Programm aendern, so dass NEXT eine zugehoerige Anweisung FOR besitzt.
2	Syntax error (Syntaxfehler) Eine Zeile enthaelt eine nicht gueltige Folge von Zeichen, wie z.B. ungleiche Anzahl von Klammern, einen falsch geschriebenen Befehl, eine falsch geschriebene Anweisung, eine ungueltige Interpunktion, oder die Daten einer Anweisung DATA sind nicht vom gleichen Typ (numerisch oder Zeichenkette) wie die Variable in einer Anweisung READ oder es wurde ein reserviertes Wort als Variablenname benutzt. Wenn dieser Fehler auftritt, zeigt das BASIC-Korrekturprogramm automatisch die fehlerhafte Zeile an. Die Zeile des Programms muss verbessert werden.
3	RETURN without GOSUB (RETURN ohne GOSUB) Eine Anweisung RETURN benoetigt vorher eine Anweisung GOSUB. Programm korrigieren. Wahrscheinlich fehlt eine Anweisung STOP oder END vor dem Unterprogramm, die bewirkt, dass das Programm nicht in den Unterprogrammcode "faellt".
4	Out of data (Zuwenig Daten) Eine Anweisung READ versucht, mehr Daten zu lesen als in der Anweisung DATA angegeben sind. Das Programm korrigieren, so dass genug Konstanten in den Anweisungen DATA fuer alle Anweisungen READ des Programms zur Verfuegung stehen.

5 Illegal function call (Ungueltiger Funktionsaufruf)
Ein sich nicht im gueltigen Bereich befindlicher Parameter wurde an eine Systemfunktion uebergeben. Der Fehler kann auch als Ergebnis der folgenden Punkte auftreten:

- Ein negativer oder zu grosser Index.
- Versuch, eine negative Zahl mit einer Zahl zu potenzieren, die nicht ganzzahlig ist.
- Funktion USR Aufruf bevor die Startadresse mit DEF USR definiert ist.
- Eine negative Satznummer fuer GET oder PUT (Datei).
- Ein ungueltiges Argument fuer eine Funktion oder Anweisung.
- Versuch ein geschuetztes BASIC-Programm aufzulisten oder zu veraendern.
- Versuch, Zeilennummern zu loeschen, die nicht existieren.

Korrigieren des Programms; siehe entsprechenden Befehl.

6 Overflow (Ueberlauf)
Der Bereich einer Zahl ist zu gross, um im BASIC-Zahlenformat dargestellt werden zu koennen. Ueberlauf fuer ganze Zahlen stoppt die Ausfuehrung, anderen falls wird die groesste in der Maschine darstellbare Zahl mit dem richtigen Vorzeichen als Ergebnis ueber geben und die Ausfuehrung fortgesetzt. Nur ein ganz zahliger Ueberlauf kann abgefragt werden.
Um einen Ueberlauf fuer ganze Zahlen zu korrigieren, muessen kleinere Zahlen verwendet werden oder zu Variablen mit einfacher oder doppelter Genauigkeit uebergegangen werden.

Hinweis: Ist eine Zahl zu klein, um im Zahlenformat von BASIC dargestellt zu werden, erhaelt man eine Unterlaufbedingung. Wenn dies auftritt, ist das Er gebnis Null, und die Ausfuehrung wird ohne einen Fehler fortgesetzt.

7 Out of memory (Hauptspeicher reicht nicht aus)
Ein Programm ist zu gross, enthaelt zu viele FOR-Schleifen oder GOSUB's, zu viele Variablen, Aus druecke, die zu kompliziert sind, oder komplexes Malen.

Mit CLEAR am Anfang des Programms kann der Stapel- oder Hauptspeicherbereich vergroessert werden.

8 Undefined line number (Nicht definierte Zeilennummer)
Eine Zeilenreferenz in einer Anweisung oder in einem Befehl spricht eine Zeile an, die sich nicht im Programm befindet.

Die Zeilennummern im Programm pruefen und die rich tige Zeilennummer benutzen.

9 Subscript out of range (Index ausserhalb des Be reichs)
Ein Feldelement wurde mit einem Index ausserhalb der Dimensionen des Feldes oder mit einer falschen Anzahl von Indizes angesprochen.
Benutzung der Feldvariablen ueberpruefen. Vielleicht wurde an eine Variable ein Index angefuegt, die keine Feldvariable ist, oder eine eingebaute Funktion wurde falsch codiert.

10 Duplicate Definition (Doppelte Definition)
Es wurde versucht, die Groesse des gleichen Feldes zweimal zu definieren. Dies kann auf verschiedene Weise passieren:

- Das gleiche Feld wurde in zwei Anweisungen DIM definiert.

- Das Programm fuehrt eine Anweisung DIM fuer ein Feld aus, nachdem die Standarddimension von 10 fuer dieses Feld gesetzt wurde.

- Das Programm erkennt eine Anweisung OPTION BASE, nachdem ein Feld entweder durch eine Anweisung DIM oder durch die Standardannahme dimensioniert wurde.

Die Anweisung OPTION BASE muss an eine Stelle im Programm gesetzt werden, so dass sie ausgefuehrt wird bevor irgendein Feld benutzt wird; oder das Programm muss korrigiert werden, so dass jedes Feld nur einmal definiert wird.

11 Division by zero (Division durch Null)
In einem Ausdruck wurde versucht, durch Null zu divi dieren, oder es wurde versucht, Null mit einer nega tiven Zahl zu potenzieren.

Es ist nicht noetig, diese Bedingung zu korrigieren, da das Programm in der Ausfuehrung fortfaehrt. Das Ergebnis der Division ist die groesstmoegliche dar stellbare Zahl mit dem Vorzeichen der Zahl, die divi diert werden sollte (Dividend); oder eine positive groesstmoegliche darstellbare Zahl als Ergebnis der Potenzierung. Dieser Fehler kann nicht abgefragt werden.

12 Illegal direct (Ungueltiger Direktmodus)
Es wurde versucht, eine Anweisung im Direktmodus einzugeben, fuer die der Direktmodus ungueltig ist (wie z.B. DEF FN).

Die Anweisung muss als Teil einer Programmzeile ein gegeben werden.

13 Typ mismatch (Keine Typuebereinstimmung)
Es wurde ein Zeichenkettenwert, anstelle eines nume rischen Wertes eingegeben oder es wurde ein numeri scher Wert anstelle eines Zeichenkettenwertes ange geben. Dieser Fehler kann auch auftreten, wenn man versucht, Variablen verschiedenen Typs auszutauschen, wie z.B. Variablen einfacher mit Variablen doppelter Genauigkeit.

14 Out of string space (Platz fuer Zeichenketten zu Ende)
BASIC ordnet den Platz fuer Zeichenketten dyna misch zu, bis kein Hauptspeicherplatz mehr zur Ver fuegung steht. Die Nachricht bedeutet, dass ausge loest durch Zeichenkettenvariablen, BASIC mehr als den verbleibenden Hauptspeicherplatz benoetigt, nach dem alles geloescht wurde, was geloescht werden konnte.

15 String too long (Zeichenkette zu lang)
Es wurde versucht, eine Zeichenkette zu erstellen, die laenger als 255 Zeichen ist.

Versuchen, die Zeichenkette in kleinere Zeichenketten aufzuteilen.

16 String formula too complex (Formel fuer Zeichenkette zu komplex)
Ein Zeichenkettenausdruck ist zu lang oder zu kom plex.

Der Ausdruck sollte in kleinere Ausdruecke zerlegt werden.

17 Can't continue (Programmausfuehrung kann nicht fort gesetzt werden)
Es wurde versucht, mit CONT die Ausfuehrung eines Programms fortzusetzen, das:

- durch einen Fehler gestoppt wurde;

- waehrend einer Ausfuehrungsunterbrechung veraendert wurde;

- nicht existiert.

Das Programm muss geladen und mit RUN ausgefuehrt werden.

18 Undefined user function (Nicht definierte Benutzer funktion)
Eine Fnktion wurde aufgerufen, bevor sie mit der Anweisung DEF FN definiert wurde.

Das Programm muss die Anweisung DEF FN ausfuehren, bevor die Funktion benutzt werden kann.

19 No RESUME (Keine Wiederaufnahme des Programms)
Das Programm verzweigt zu einer aktiven Fehlerbehand lungsroutine als Ergebnis einer Fehlerbedingung oder einer Anweisung ERROR. In dieser Routine fehlt eine Anweisung RESUME. (Das physische Ende des Programms wurde in der Fehlerbehandlungsroutine gefunden.)

RESUME muss in die Fehlerbehandlungsroutine einge fuegt werden, um mit der Programmausfuehrung fortfah ren zu koennen. Man kann in die Fehlerbehandlungsrou tine auch die Anweisung ON ERROR GOTO 0 einfuegen, so dass BASIC fuer jeden nicht gelesenen Fehler die Nachricht anzeigt.

20 RESUME without error (RESUME ohne Fehler)
Das Programm hat eine Anweisung RESUME gefunden, ohne durch einen Fehler dorthin gekommen zu sein. In die Fehlerbehandlungsroutine sollte nur verzweigt werden, wenn ein Fehler auftrat oder eine Anweisung ERROR ausgefuehrt wird.
Wahrscheinlich muss eine Anweisung STOP oder END vor der Fehlerbehandlungsroutine eingefuegt werden, um zu verhindern, dass das Programm in die Fehlerbehandlungsroutine "faellt".

22 Missing operand (Fehlender Operand)
Ein Ausdruck enthaelt einen Operator, wie z.B. * (Stern) oder OR, dem kein Operand folgt.

Alle benoetigten Operanden muessen in den Ausdruck eingefuegt werden.

23 Line buffer overflow (Zeilenpufferueberlauf)
Es wurde versucht, eine Zeile einzugeben, die aus zu vielen Zeichen besteht.
Einzeilige Mehrfachanweisungen in mehrere Zeilen aufteilen. Es koennen auch Zeichenkettenvariablen anstelle von Konstanten benutzt werden, wo dies moeglich ist.

24 Device Timeout (Einheitenzeitsperre)
BASIC erhielt innerhalb einer vorgegebenen Zeitspanne keine Information von einer Ein-/Ausgabeeinheit.
Fuer Datenfernverarbeitungsdateien bedeutet diese Nachricht, dass ein oder mehrere Signale, die mit OPEN "COM... gesetzt wurden, in einer angegebenen Zeitperiode nicht gefunden wurden.

Operation wiederholen.

25 Device Fault (Einheitenfehler)
Ein Hardware-Fehler wurde von einem Anschluss ange zeigt.

Diese Nachricht kann auch auftreten, wenn Daten in eine Datenfernverarbeitungsdatei uebertragen werden. In diesem Fall wird damit angezeigt, dass ein oder mehrere Signale, die getestet werden (angegeben mit der Anweisung OPEN "COM...), nicht in der vorgegebe nen Zeitspanne gefunden wurden.

26 FOR without NEXT (FOR ohne NEXT)
FOR wurde ohne ein zugehoeriges NEXT gefunden. Das heisst, eine FOR-Schleife war aktiv, als das physische Ende des Programms erreicht wurde.

Das Programm durch Einfuegen einer Anweisung NEXT verbessern.

27 Out of Paper (Kein Druckerpapier)
Im Drucker ist kein Papier oder er ist nicht einge schaltet.

Papier einlegen (falls noetig), die Verbindung des Druckers ueberpruefen oder den Drucker einschalten. Danach mit dem Programm fortfahren.

29 WHILE without WEND (WHILE ohne WEND)
Einer Anweisung WHILE fehlt die zugehoerige Anweisung WEND. Das heisst, WHILE war noch aktiv, als das physische Ende des Programms erreicht wurde.

Das Programm durch Einfuegen einer Anweisung WEND fuer jedes WHILE korrigieren.

30 WEND without WHILE (WEND ohne WHILE)
Eine Anweisung WEND wurde gefunden, bevor die zugehoerige Anweisung WHILE ausgefuehrt wurde.

Das Programm so verbessern, dass fuer jede Anweisung WEND eine Anweisung WHILE existiert.

50 FIELD overflow (FELD-Ueberlauf)
Eine Anweisung FIELD versucht, mehr Bytes zuzuordnen als fuer die Satzlaenge in einer Datei fuer wahl freien Zugriff in der Anweisung OPEN angegeben waren, oder es wurde das Ende des FIELD-Puffers gefunden, waehrend eine sequentielle Ein-/Ausgabe (PRINT#, WRITE#, INPUT#) zu einer Datei mit wahlfreiem Zugriff erfolgte.

Die Anweisungen OPEN und FIELD sind zu ueberpruefen und beide auf Uebereinstimmung zu bringen. Bei se quentieller Ein-/Ausgabe zu einer Datei mit wahl freiem Zugriff darf die Laenge der gelesenen oder geschriebenen Daten die Satzlaenge der Datei fuer wahlfreien Zugriff nicht ueberschreiten.

51 Internal error (Interner Fehler)
In BASIC trat ein interner Fehler auf.

Diskette neu kopieren. Die Hardware pruefen und die Operation wiederholen. Im erneuten Fehlerfalle das Serviceunternehmen ueber die Fehlerbedingungen infor mieren.

52 Bad file number (Falsche Dateinummer)
Eine Anweisung benutzt eine Dateinummer einer Datei, die nicht eroeffnet ist, oder die Dateinummer befin det sich nicht in dem moeglichen Dateinummernbereich, der bei der Initialisierung angegeben wurde; oder der Einheitenname in der Dateispezifikation ist zu lang oder ungueltig oder der Dateiname war zu lang oder ungueltig.

Sicherstellen, dass die gewuenschte Datei eroeffnet war oder die Dateinummer richtig in die Anweisung eingefuegt wurde. Auch auf gueltige Dateispezifika tion ueberpruefen (siehe unter "Namensgebung fuer Dateien" in Kapitel 6).

53 File not found (Datei nicht gefunden)
Mit LOAD, KILL, NAME, FILES oder OPEN wurde eine Datei angesprochen, die nicht auf der Diskette in der angegebenen Einheit steht.

Ueberpruefen, ob sich die richtige Diskette in der angegebenen Einheit befindet und ob die Dateispezifi kation stimmt. Danach die Operation wiederholen.

54 Bad file mode (Falscher Dateityp)
Mit PUT oder GET wurde versucht, eine sequentielle Datei oder eine geschlossene Datei anzusprechen, oder ein OPEN mit einem anderen Dateityp als Eingabe, Ausgabe, Hinzufuegen oder wahlfreier Zugriff auszu fuehren.

Sicherstellen, dass die Anweisung OPEN richtig einge
geben und ausgefuehrt wurde. GET und PUT benoetigen
eine Datei fuer wahlfreien Zugriff.

Dieser Fehler tritt auch auf, wenn man versucht, eine
Datei, die nicht im ASCII-Format vorliegt, zu mi
schen. In diesem Fall muss man sicherstellen, dass
die richtige Datei gemischt wird. Wenn noetig, das
Programm laden und mit der Auswahl A speichern.

55 File already open (Datei schon eroeffnet)
Es wurde versucht, eine Datei fuer sequentielle
Ausgabe oder fuer sequentielles Hinzufuegen zu er
oeffnen, und die Datei war schon eroeffnet; oder es
wurde versucht, KILL fuer eine schon eroeffnete Datei
zu benutzen.

Sicherstellen, dass nur ein OPEN fuer eine Datei
ausgefuehrt wird, wenn sie sequentiell beschrieben
werden soll. Die Datei abschliessen, bevor KILL be
nutzt wird.

57 Device I/O Error (Einheitenein-/-ausgabefehler)
Ein Fehler passierte bei einer Einheitenein-/-ausga
beoperation. DCP kann diesen Fehler nicht beheben.
Bei Empfang von Datenfernverarbeitungsdaten kann
dieser Fehler durch Ueberlauf, Datenfernverarbei-
tungsrahmen, Unterbrechung oder Paritaetsfehler auf
treten. Werden Daten mit sieben oder weniger Daten-
Bits empfangen, wird das achte Bit in das fehlerhafte
Byte eingeschaltet.

58 File already exists (Die Datei existiert schon)
Eine in der Anweisung NAME angegebener Dateiname
existiert schon auf der Diskette.

Den Befehl NAME mit einem anderen Namen wiederholen.

61 Disk full (Diskette voll)
Der Speicher auf der Diskette ist vollstaendig be
nutzt. Dateien werden abgeschlossen, wenn dieser
Fehler auftritt.

Befinden sich Dateien auf der Diskette, die nicht
mehr benoetigt werden, sollten sie geloescht werden;
oder es sollte eine neue Diskette benutzt werden.
Danach die Operation wiederholen oder das Programm
neu laufen lassen.

62 Input past end (Eingabe nach logischem Ende)
Dies ist ein Dateiendefehler. Eine Eingabeanweisung
wurde fuer eine leere Datei ausgefuehrt oder nachdem
alle Daten einer sequentiellen Datei schon gelesen
waren.
Um diesen Fehler zu vermeiden, sollte die Funktion
EOF benutzt werden, um das Ende der Datei abzufangen.

Der gleiche Fehler wird angezeigt, falls versucht
wird, von einer Datei zu lesen, die fuer Ausgabe oder
Hinzufuegen eroeffnet wurde. Soll von einer sequen
tiellen Ausgabe- (oder Hinzufuege-) Datei gelesen
werden, muss sie geschlossen und wieder fuer Eingabe
eroeffnet werden.

63 Bad secord number (Falsche Satznummer)
In einer Anweisung PUT oder GET ist die Satznummer entweder groesser als die maximal erlaubte oder gleich Null.

Die Anweisung PUT oder GET korrigieren, damit eine gueltige Satznummer angesprochen wird.

64 Bad file name (Falscher Dateiname)
Eine ungueltige Form wurde fuer den Dateinamen mit BLOAD, BSAVE, KILL, NAME, OPEN oder FILES benutzt.

In Kapitel 6 unter "Namensgebung fuer Dateien" nach sehen, wie gueltige Dateinamen aussehen muessen, und den fehlerhaften Dateinamen korrigieren.

66 Direct Statement in file (Anweisung fuer direkten Modus in der Datei)
Eine Anweisung fuer direkten Modus wurde gefunden, waehrend ein Programm mit LOAD oder CHAIN im ASCII-Format geladen wurde. LOAD oder CHAIN werden beendet.

Die ASCII-Datei darf nur Anweisungen enthalten, denen eine Zeilennummer vorausgeht: Der Fehler kann auch auftreten, weil sich ein Zeichen fuer Zeilenvorschub im Eingabestrom befindet.

67 Too many files (Zu viele Dateien)
Es wurde versucht, eine neue Datei zu erstellen (mit Hilfe von SAVE oder OPEN), als alle Bibliotheks-eintragungen der Diskette voll waren oder die Datei spezifikation ungueltig war.

Ist die Dateispezifikation in Ordnung, muss eine neue formatierte Diskette verwendet und die Operation wiederholt werden.

68 Device Unavailable (Einheit nicht verfuegbar)
Es wurde versucht, eine Datei in einer Einheit zu eroeffnen, die nicht vorhanden ist. Entweder fehlt die Hardware, um die Einheit zu unterstuetzen (wie z.B. Druckeranschluesse fuer den zweiten oder dritten Drucker) oder die Einheit wurde inaktiviert (Es wurde z.B. /C:0 beim Laden von BASIC benutzt. Dadurch wuer den die Datenfernverarbeitungseinheiten inaktiviert).

Die Einheit muss richtig installiert sein. Falls noetig, zu DCP zurueckverzweigen, wo der BASIC-Befehl erneut eingegeben werden kann.

69 Communication buffer overflow (Ueberlauf des Daten fernverarbeitungspuffers)
Eine Eingabeanweisung fuer Datenfernverarbeitung wur de ausgefuehrt, aber der Eingabepuffer war schon voll.

Die Anweisung ON ERROR muss benutzt werden, um die Eingabe zu wiederholen, falls diese Bedingung auf tritt. Nachfolgende Eingaben versuchen, diesen Fehler zu loeschen, ausser die Zeichen werden weiterhin schneller empfangen, als das Programm sie verarbeiten kann. Es gibt verschiedene Loesungen:

- Die Groesse des Datenfernverarbeitungspuffers mit Hilfe der Auswahl /C: erweitern, wenn mit BASIC gestartet wird.

- Ein "Hand-Shaking"-Protokoll mit dem anderen Com puter implementieren, um ihm mitzuteilen, so lange nicht zu senden, bis die empfangenen Daten verarbeitet sind.

- Eine niedrigere Uebertragungsrate fuer Senden und Empfangen benutzen.

70 Disk Write Protect (Diskettenschreibschutz)
Es wurde versucht, eine Diskette mit Disketten schreibschutz zu beschreiben.

Ueberpruefen, ob die richtige Diskette benutzt wird. Wenn dies der Fall ist, den Schreibschutz entfernen und die Operation wiederholen.

Dies kann auch ein Hardware-Fehler sein.

71 Disk not Ready (Diskette nicht bereit)
Die Diskettenverriegelung ist offen oder in der Ein heit befindet sich keine Diskette.

Die richtige Diskette in das Laufwerk einlegen und mit dem Programm fortfahren.

72 Disk Media Error (Diskettenfehler)
Die Steueranschlusskarte entdeckt einen Hardware-Fehler oder einen Diskettenfehler. Im allgemeinen bedeutet dies, dass eine Diskette fehlerhaft ist.

Alle sich auf der Diskette befindlichen Dateien auf eine neue Diskette kopieren und die fehlerhafte Dis kette neu formatieren. Ist das Formatieren nicht moeglich, sollte die Diskette nicht mehr benutzt werden.

74 Rename across disks (Umbenennen auf Diskette)
Es wurde versucht, eine Datei umzubenennen, aber die falsche Diskette wurde angegeben. Die Umbenennung findet nicht statt.
Wenn **RENAME** benutzt wird, muss fuer den alten und den neuen Dateinamen das gleiche Laufwerk angegeben wer den, ausser der DCP-Befehl ASSIGN ist aktiviert. In diesem Fall kann der Laufwerksname unterschiedlich sein, es muss jedoch dieselbe physische Einheit ange geben werden.

75 Path/file access error (Pfad/Datei-Zugriffsfehler)
Waehrend einer OPEN-, RENAME-, MKDIR-, CHDIR- oder RKDIR-Operation wurde der Versuch unternommen, einen Pfad oder Dateinamen zu einer Datei zu benutzen, auf die nicht zugegriffen werden kann. Zum Beispiel wurde versucht, ein Inhaltsverzeichnis oder einen Daten traeger-Kennsatz zu eroeffnen, oder eine Datei, die nur fuer Lesen definiert war, fuer Schreiben zu er oeffnen, oder das laufende Verzeichnis zu loeschen. Die Operation wird nicht beendet.

76 Path not found (Pfad nicht gefunden)
Waehrend einer OPEN-, MKDIR-, CHDIR- oder RMDIR-Operation kann DCP den angegebenen Pfad nicht finden. Die Operation wird nicht beendet.

- Unprintable error (Nichtdruckbarer Fehler)
  Fuer die existierende Fehlerbedingung ist keine Fehlernachricht verfuegbar. Dies passiert gewoehnlich durch eine Anweisung ERROR mit einem nicht definierten Fehlercode.
  Im Programm pruefen, ob alle erstellten Fehlercodes auch bearbeitet werden.

- Incorrect DCP-Version (Falsche DCP-Version)
  Der gerade eingegebene Befehl gehoert zu einer anderen DCP-Version.

- You cannot SHELL to Basic (SHELL kann nicht zu BASIC durchgefuehrt werden).
  BASIC kann nicht als Kind-Prozess ausgefuehrt werden.

- Can't continue after SHELL (Programm kann nach SHELL nicht fortgesetzt werden)
  Ein Kind-Prozess, der durch die Anweisung SHELL ausgefuehrt wurde, sollte nie beendet werden und resident bleiben, da sonst fuer BASIC nicht genuegend Speicherplatz zur Verfuegung steht und die Programmausfuehrung angehalten wird.

AnlageD

ASCII-Zeichencodes

In der folgenden Tabelle sind alle ASCII-Codes (dezimal, hexa dezimal und ihre zugehoerigen Zeichen) aufgelistet.
Diese Zeichen koennen mit Hilfe von PRINT CHR$(n) angezeigt werden, wobei n der ASCII-Code ist.
In der Tabelle sind die Standardinterpretationen der ASCII-Codes 0 bis 31 aufgefuehrt. Dies sind nicht darstellbare Zeichen und werden gewoehnlich fuer Steuerfunktionen oder Datenfernverarbeitung benutzt.
Jedes dieser Zeichen kann ueber die Tastatur eingegeben wer den, indem man die Taste "ALT" betaetigt und niederhaelt und dann die Ziffern fuer den ASCII-Code auf der Zehnertastatur eingibt. Dabei muss beachtet werden, dass einige dieser Codes eine besondere Bedeutung fuer das BASIC-Korrekturprogramm haben - das BASIC-Korrekturprogramm benutzt seine eigene In terpretation fuer diese Codes und kann die hier aufgelisteten Sonderzeichen nicht anzeigen.
Hinweis:
Die Zeichen fuer die ASCII-Codes 128 bis 255 koennen nur im Textmodus dargestellt werden. Siehe Anweisung SCREEN in diesem Handbuch und Abschnitt "Benutzung des Bildschirms" im BASIC-Handbuch.
ErweiterteCodes

Fuer gewisse Tasten und Tastenkombinationen, die nicht im Standard ASCII-Code dargestellt werden koennen, wird ein er weiterter Code von der Systemvariablen INKEY$ uebergeben. Eine Leerstelle (ASCII-Code 000) wird als erstes Zeichen einer Zwei-Zeichen-Zeichenkette uebergeben. Wird INKEY$ eine Zei chenkette aus zwei Zeichen empfangen, sollte man zurueckver zweigen und das zweite Zeichen pruefen, um zu bestimmen, welche Taste betaetigt wurde. Im allgemeinen, aber nicht im mer, ist dieser zweite Code der Pruefcode der Primaertaste, die betaetigt wurde. Die ASCII-Codes (dezimal) fuer dieses zweite Zeichen und die zugehoerigen Tasten sind unten aufge listet.

Zweiter Code	Bedeutung
3	(Leerstelle)NUL
15	(Umschalten Tabulator)| <--
16-25	ALT-Q, W, E, R, T, Y, U, I, O, P
30-38	ALT-A, S, D, F, G, H, J, K, L
44-50	ALT-Z, X, C, V, B, N, M
59-68	Funktionstasten F1 bis F10 (falls inaktiviert als Funktionstasten)
71	HOME
72	Kursor nach oben
73	PG UP
75	Kursor nach links
77	Kursor nach rechts
79	END
80	Kursor nach unten
81	PG DN
82	INS
83	DEL
84-93	F11-F20 (Shift-F1 bis F10)
94-103	F21-F30 (CTRL-F1 bis F10)
104-113	F31-F40 (ALT-F1 bis F10)
114	CTRL-PRTSC
115	CTRL-Kursor nach links (vorheriges Wort)
116	CTRL-Kursor nach rechts (naechstes Wort)
117	CTRL-END
118	CTRL-PG DN

119	CTRL-HOME
120-131	ALT-1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, =
132	CTRL-PG UP

ANLAGE E

HexadezimaleUmwandlungstabelle

Hex	Dezimal	Hex	Dezimal
1	1	10	16
2	2	20	32
3	3	30	48
4	4	40	64
5	5	50	80
6	6	60	96
7	7	70	112
8	8	80	128
9	9	90	144
A	10	A0	160
B	11	B0	176
C	12	C0	192
D	13	D0	208
E	14	E0	224
F	15	F0	240
100	256	1000	4096
200	512	2000	8192
300	768	3000	12288
400	1024	4000	16384
500	1280	5000	20480
600	1536	6000	24576
700	1792	7000	28672
800	2048	8000	32768
900	2304	9000	36864
A00	2560	A000	40960
B00	2816	B000	45056
C00	3072	C000	49152
D00	3328	D000	53248
E00	3584	E000	57344
F00	3840	F000	61440

Hex	Dezimal	Hex	Dezimal
1	1	10	16
2	2	20	32
3	3	30	48
4	4	40	64
5	5	50	80
6	6	60	96
7	7	70	112
8	8	80	128
9	9	90	144
A	10	A0	160
B	11	B0	176
C	12	C0	192
D	13	D0	208
E	14	E0	224
F	15	F0	240
100	256	1000	4096
200	512	2000	8192
300	768	3000	12288
400	1024	4000	16384
500	1280	5000	20480
600	1536	6000	24576
700	1792	7000	28672

800	2048	8000	32768
900	2304	9000	36864
A00	2560	A000	40960
B00	2816	B000	45056
C00	3072	C000	49152
D00	3328	D000	53248
E00	3584	E000	57344
F00	3840	F000	61440

ANLAGE F

Mathematische Funktionen

Funktionen, die nicht im BASIC des Personalcomputers eingebaut sind, koennen wie folgt berechnet werden:

Funktion	Aequivalent
Logarithmus zur Basis B	LOG(X) LOG(X)/LOG(B)
Sekans	SEC(X) = 1/COS(X)
Cosekans	CSC(X) = 1/SIN(X)
Cotangens	COT(X) = 1/TAN(X)
Arcussinus	ARCSIN(X) = ATN(X/SQR(1-X*X))
Arcuscosinus	ARCCOS(X) = 1.570796-ATN(X/SQR(1-X*X))
Arcussekans	ARCSEC(X) = ATN(SQR(X*X-1))+(X<0)*3.141593
Arcuscosekans	ARCCSC(X) = ATN(1/SQR(X*X-1))+(X<0)*3.141593
Arcuscotangens	ARCCCOT(X) = 1.57096-ATN(X)
Sinus Hyperbolikus	SINH(X) = (EXP(X)-EXP(-X))/2
Cosinus Hyperbolikus	COSH(X) = (EXP(X)+EXP(-X))/2
Tangens Hyperbolikus	TANH(X) = (EXP(X)-EXP(-X))/(EXP(X)+EXP(-X))
Sekans Hyperbolikus	SECH(X) = 2/(EXP(X)+EXP(-X))
Cosekans Hyperbolikus	CSCH(X) = 2/(EXP(X))-EXP(-X))
Cotangens Hyperbolikus	COTH(X) = (EXP(X)+EXP(-X))/(EXP(X)-EXP(-X))
Arcussinus Hyperbolikus	ARCSINH(X) = LOG(X+SQR(X*X+1))
Arcuscosinus Hyperbolikus	ARCCOSH(X) = LOG(X+SQR(X*X-1))
Arcustangens Hyperbolikus	ARCTANH(X) = LOG((1+X)/(1-X))/2
Arcussekans Hyperbolikus	ARCSECH(X) = LOG((1+SQR(1-X*X))/X
Arcuscosekans Hyperbolikus	ARCCSCH(X) = LOG((1+SGN(X)*SQR(1+X*X))/X
Arcuscotangens Hyperbolikus	ARCCOTH(X) = LOG((X+1)/(X-1))/2

Werden diese Funktionen benutzt, ist es ein guter Weg, sie mit der Anweisung DEF FN zu codieren. Statt die Formel fuer den Arcussinus Hyperbolikus jedesmal, wenn er benoetigt wird, zu codieren, koennte man dies folgendermassen tun:

```
DEF FNARCSINH(X) = LOG(X+SQR(X*X+1))
```

Danach koennte man die Funktionen wie folgt aufrufen:

```
FNARCSINH(Y)
```

Anlage G

InterpretationspeziellerZeichendurchdenBildschirm

ASCII	Zeichen	Wirkung
0	(leer)	
1		
2		
3		
4		
5		
6		
7	(Alarm)	
8		
9	(Tabulator)	
10	(LF)	Kursor eine Zeile nach unten
11		
12	(FF)	Bildschirm loeschen und Kursor an den Anfang
13	(CR)	Kursor an Zeilenanfang der naechsten Zeile
14		
15		
16		
17		
18	|	
19	!!	
20		
21		
22	-	
23	|	
24	|	
25	|	
26	-->	
27	<--	
28		Kursor nach rechts
29		Kursor nach links
30		Kursor nach oben
31		Kursor nach unten

SteuerzeichenfuerDruckerK6313/14

Die Steuerung des Druckers erfolgt auf Basis des ASCII-Codes.
Spezielle Steuerzeichenfolgen werden mit Escape (27) eingelei
tet.

Beispiel: Wirkung:

```
LPRINT CHR$(27);"E"      Einschalten Fettschrift.
LPRINT CHR$(12)          Formularvorschub
```

Steuerzeichen	Wirkung
SO (14)	Einschalten Sperrschrift
SI (15)	Einschalten komprimierte Schrift
DC2 (18)	Ausschalten komprimierte Schrift
DC4 (20)	Ausschalten Sperrschrift
ESC E	Einschalten Fettschrift
ESC F	Ausschalten Fettschrift
ESC G	Einschalten Doppeldruck
ESC H	Ausschalten Doppeldruck
ESC S	Einschalten Hoch- oder Tiefschrift
ESC T	Ausschalten Hoch- oder Tiefschrift
ESC W	Ein- oder Ausschalten Sperrschrift
ESC -	Ein- oder Ausschalten Unterstreichmodus
ESC K	Einzelpunktmodus 480 Sprossen pro 8 Zoll
ESC L	Einzelpunktmodus 960 Sprossen pro 8 Zoll
ESC Y	Einzelpunktmodus 960 Sprossen pro 8 Zoll
ESC Z	Einzelpunktmodus 1920 Sprossen pro 8 Zoll
ESC 0/	Setzen des Zeilenabstandes auf 1/8"
ESC 1	Setzen des Zeilenabstandes auf 7/72"
ESC 2	Start des variablen Zeilenabstandes
ESC 3	Setzen des Zeilenabstandes auf n/216"
ESC A	Voreinstellen eines variablen Zeilenabstandes von n/72"
LF (10)	Zeilenvorschub
VT (11)	Vertikaltabulation
FF (12)	Formularvorschub
ESC J	Ausfuehrung eines Zeilenvorschubes von n/216"
ESC N	Einstellen Formularendezeile
ESC O	Ausschalten Formularendezeile
NUL (0)	Ende Tabulator setzen
HT (9)	Horizontal-Tabulation
CR (13)	Druckposition auf Zeilenanfang setzen
ESC C	Einstellen Formularlaenge (n Zeilen)
ESC C0/	Einstellen Formularlaenge (n Zoll)
ESC D	Setzen von Horizontaltabulations-Marken
CAN (24)	Loeschen des Druckpuffers
BEL (8)	Summer
ESC 8	Ausschalten Papierendekontrolle
ESC 9	Einschalten Papierendekontrolle
ESC 6	Auswahl Zeichensatz 2
ESC 7	Auswahl Zeichensatz 1
ESC U	Einstellen uni- und bidirektionaler Druck
ESC <	Einzeiliger unidirektionaler Druck 1 Zeile von links beginnend
ESC x	Ein- oder Ausschalten des NLQ-Modus

Anlage H

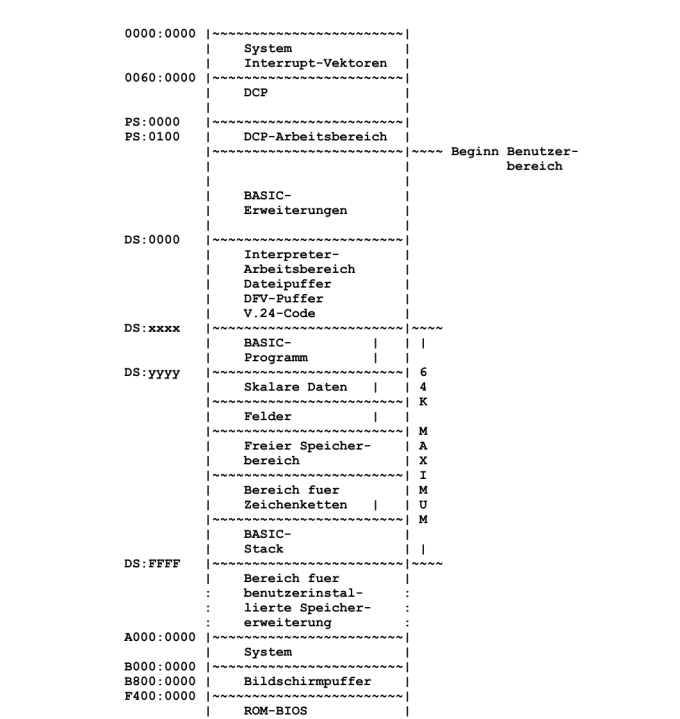
TechnischeInformationen

1. Hauptspeicherbelegung
2. Speicherung von Variablen
3. Tastaturpuffer
4. Suchordnung fuer Anschluesse
5. Umschalten der Bildschirme

1.Hauptspeicherbelegung

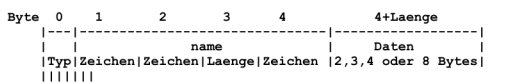
Die Adressen stehen in der folgenden Hexadezimalform:

Segment:Relativzeiger.



2.SpeicherungvonVariablen

Skalarvariablen werden im BASIC-Datenbereich wie folgt gespei
chert:



Typ
Gibt den Variablentyp an:

2 Ganzzahlig
3 Zeichenkette
4 Einfache Genauigkeit
8 Doppelte Genauigkeit

Name
Ist der Name der Variablen. Die ersten zwei Zeichen des Namens
sind in Byte 1 und Byte 2 gespeichert. Byte 3 sagt aus, wie
viel weitere Zeichen sich im Variablennamen befinden. Diese
zusaetzlichen Zeichen werden ab Byte 4 gespeichert.
Dies bedeutet, dass jeder Variablenname mindestens drei Bytes
belegt. Ein Name aus ein oder zwei Zeichen benoetigt exakt
drei Bytes. Ein Name aus x Zeichen benoetigt x+1 Bytes.

Daten
Folgt dem Namen der Variablen und kann zwei, drei, vier oder
acht Bytes lang sein (wie unter Typ beschrieben). Der Wert,
der von der Funktion VARPTR uebergeben wird, zeigt auf diese
Daten.

Fuer Zeichenkettenvariablen bedeutet Daten die Zeichenketten
beschreibung:

. Das erste Byte der Zeichenkettenbeschreibung enthaelt
  die Laenge der Zeichenkette (0 bis 255).

. Die letzten zwei Bytes der Zeichenkettenbeschreibung
  enthalten die Adresse der Zeichenkette im BASIC-Datenbe
  reich (den Relativzeiger in das Standardsegment). Adres
  sen werden mit dem niedrigen Byte zuerst und mit hoehe
  ren Byte danach gespeichert:

  - Das zweite Byte der Zeichenkettenbeschreibung ent
    haelt also das niedrige Byte des Relativzeigers.

  - Das dritte Byte der Zeichenkettenbeschreibung ent
    haelt das hohe Byte des Relativzeigers.

Fuer numerische Variablen enthaelt Daten den aktuellen Wert
der Variablen:

. Ganzzahlige Werte werden in zwei Byte gespeichert, das
  niedrige Byte zuerst und das hohe Byte danach.

. Werte einfacher Genauigkeit werden in vier Bytes gespei
  chert, und zwar in der internen Gleitkommabinaerdarstel
  lung von BASIC.

. Werte doppelter Genauigkeit werden in acht Bytes gespei
  chert, und zwar in der internen Gleitkommabinaerdarstel
  lung von BASIC.

3.Tastaturpuffer

Zeichen, die ueber die Tastatur eingegeben werden, werden im Tastaturpuffer gespeichert, bis sie verarbeitet werden. Bis zu 15 Zeichen koennen im Puffer gespeichert werden; wird versucht, mehr als 15 Zeichen einzugeben, gibt der Computer ein Alarmzeichen.

INKEY$ liest nur ein Zeichen aus dem Tastaturpuffer, selbst wenn mehrere Zeichen vorhanden sind. Mit INPUT$ koennen mehrere Zeichen gelesen werden. Steht jedoch die angegebene Anzahl von Zeichen nicht im Puffer, wartet BASIC, bis genug Zeichen eingegeben sind.

Der Systemtastaturpuffer kann durch die folgende Programmzeile geloescht werden:

```
WHILE INKEY$ <> "": WEND
```

Diese Technik kann nuetzlich sein, wenn der Puffer geloescht werden soll, bevor der Benutzer "irgendeine Taste" betaetigen soll.

BASIC hat seinen eigenen Zeilenpuffer, in den das BASIC-Korrekturprogramm Zeichen verarbeitet, die von dem System tastaturpuffer empfangen werden. Der BASIC-Zeilenpuffer kann durch die folgende Programmzeile geloescht werden:

```
DEF SEG: POKE 106,0
```

4.Suchordnung fuer Anschluesse

Die Drucker mit Zuordnung LPT1:, LPT2: und LPT3: werden zugeordnet, wenn der Computer eingeschaltet wird. Das System prueft die Druckeranschluesse nach einer bestimmten Reihenfolge. Der erste gefundene Druckeranschluss erhaelt LPT1:, der zweite Anschluss (falls vorhanden) wird LPT2: und der dritte (falls vorhanden) wird LPT3:. Die Suchordnung ist wie folgt:

1. Schwarz/Weiss-Bildschirm und Paralleldruckeranschluss
2. Paralleldruckeranschluss
3. Paralleldruckeranschluss, der veraendert wurde, um die Basisadresse zu aendern.

Wenn ein Drucker mit Hilfe des Befehls MODE von DCP neu verbunden wird, ist diese Aenderung auch in BASIC effektiv.

Die Datenfernverarbeitungseinheiten COM1: und COM2: erhalten ihre Zuordnung aehnlich wie die Drucker.

Ihre Suchordnung ist:

1. Anschlusskarte fuer asynchrone Uebertragung
2. Eine veraenderte Anschlusskarte fuer asynchrone Uebertragung.

5.Umschalten der Bildschirme

Besitzt der Personalcomputer die Anschlusskarte fuer Farb-/ Grafikbildschirm und Schwarz/Weiss-Bildschirm mit Parallel drucker, so gibt BASIC normalerweise zum Schwarz/Weiss-Bild schirm aus. Jedoch kann man mit BASIC durch den folgenden Programmcode von einem Bildschirm zum anderen umschalten:

```
10 'Umschalten auf Schw.-weiss-BS
20 DEF SEG=0
30 POKE &H410,(PEEK(&H410)OR &H30)
40 SCREEN 0
50   WIDTH 40
60   WIDTH 80
70   LOCATE,,1,12,13

10 'Umschalten auf Farb-BS
20 DEF SEG=0
30 POKE &H410,(PEEK(&H410) AND (&HCF) OR &H10)
```

Hinweis:

Mit dieser Technik wird der Bildschirm, zu dem umgeschaltet wird, geloescht. Die Stellung des Kursor kann fuer jeden Bildschirm verschieden sein.

Anlage I

Programmiertips

Oft gibt es verschiedene Wege, etwas in BASIC zu programmie ren, um das gleiche Ergebnis zu erhalten.
Dieser Abschnitt enthaelt verschiedene Programmiertips, die Programmausfuehrungszeit zu verbessern.

ALLGEMEIN

- Anweisungen kombinieren,
wo es moeglich ist, um den Vorteil der Anweisungslaenge von 255 Zeichen auszunutzen. Beispiel:

Besser

```
100 FOR I=1 TO 10 : READ A(I) : NEXT I
```

Als

```
100 FOR I=1 TO 10
110 READ A(I)
120 NEXT I
```

- Wiederholte Berechnungen fuer Ausdruecke vermeiden.
Werden identische Berechnungen in mehreren Anweisungen ausge fuehrt, kann man diese Berechnungen einmal ausfuehren und das Ergebnis in einer Variablen speichern, bevor diese in spaete ren Anweisungen benutzt wird. Beispiel:

Besser

```
300 X=C*3+D
310 A=X+Y
320 B=X+Z
```

Als

```
310 A=C*3+D+Y
320 B=C*3+D+Z
```

Jedoch geht die Zuordnung einer Konstanten zu einer Variablen schneller, als die Zuordnung des Wertes einer Variablen zu einer anderen Variablen.

- Einfache Arithmetik benutzen.
Im allgemeinen geht eine Addition schneller als eine Multipli kation, und die Multiplikation ist schneller als eine Division oder Potenzierung.

Betrachten wir diese Beispiele:

Besser

```
250 B=A*.5
500 B=A+A
650 B=A*A*A
750 B%=A%/4
```

Als

```
250 B=A/2
500 B=A*2
650 B=A^3
750 B%=INT(A%/4)
```

- Eingebaute Funktionen benutzen.
Eingebaute Funktionen sind immer schneller, als die gleiche Moeglichkeit, in BASIC programmiert.

- Mit Bemerkungen sparen.
BASIC braucht jedesmal ein klein wenig Zeit, um eine Bemerkung zu identifizieren. Mit einem Apostroph (') Bemerkungen besser an das Ende einer Zeile anfuegen, als eine eigene Anweisung benutzen. Dadurch wird der Durchsatz verbessert und es wird Hauptspeicher eingespart, weil keine Zeilennummer benoetigt wird. Beispiel:

Besser

```
10 FOR I=1 TO 10
20 A(I)=30 'Initialisiere A
30 NEXT I
```

Als

```
10 FOR I=1 TO 10
15 'Initialisiere A
20 A(I)=30
30 NEXT I
```

- Hinweis zum BASIC.
Wenn BASIC zu einer Zeilennummer verzweigt, weiss es nicht genau, wo sich die Zeile im Hauptspeicher befindet. Deshalb muss BASIC die Zeilennummern des Programms durchsuchen, und zwar vom Beginn des Programms bis zum Auffinden der Zeilennum mer.
In einigen anderen BASICs muss dieses Suchen jedesmal durchge fuehrt werden, wenn eine Verzweigung im Programm stattfindet. In diesem BASIC wird das Suchen nur einmal durchgefuehrt. Danach ist die Verzweigung direkt. Das Programm laeuft also nicht schneller, wenn haeufig benutzte Unterprogramme an den

Beginn des Programms gelegt werden.

LOGISCHE STEUERUNG

- Die Faehigkeiten der Anweisung IF nutzen.
Durch die Benutzung von AND und OR und der Angabe ELSE koennen mehrere Anweisungen IF und zusaetzlicher Code im Programm vermieden werden. Beispiel:

Besser

```
200 IF A=B AND C=D THEN Z=12 ELSE Z=B
```

Als

```
200 IF A=B THEN GOTO 210
205 GOTO 215
210 IF C=D THEN 225
215 Z=B
220 GOTO 230
225 Z=12
230 .....
```

- Anweisungen IF so anordnen, dass die am haeufigsten auftre tende Bedingung zuerst getestet wird. Dadurch werden unnoetige Tests vermieden, z.B. wenn eine Dateneingabedatei fuer Kunden bestellungen vorliegt, die aus verschiedenen Satzarten und sehr vielen individuellen Transaktionen besteht.

Eine typische Satzgruppe sieht wie folgt aus:

Code der Satzart	Satzart
A	Kennsatz
B	Kundenname und -adresse
C	Transaktion
C	Transaktion
:	:
:	:
C	Transaktion
D	Endsatz

Besser

```
100 IF TYPE$="C" THEN 3000
110 IF TYPE$="A" THEN 1000
120 IF TYPE$="B" THEN 2000
130 IF TYPE$="D" THEN 4000
```

Als

```
100 IF TYPE$="A" THEN 1000
110 IF TYPE$="B" THEN 2000
120 IF TYPE$="C" THEN 3000
130 IF TYPE$="D" THEN 4000
```

Wuerden 100 Gruppen mit jeweils zehn Transaktionen pro Gruppe vorliegen, wuerden dadurch, dass der Test der Transaktionen an den Anfang der Liste gelegt wird, 1800 IF-Anweisungen weniger ausgefuehrt.

Im folgenden wird ein weiteres Beispiel angezeigt, in dem Anweisungen IF so angeordnet werden, dass so wenig Tests wie moeglich stattfinden:

Besser

```
200 IF A<>1 THEN 250
210 IF B=1 THEN X=0
220 IF B=2 THEN X=1
230 IF B=3 THEN X=2
240 GOTO 280
250 IF B=1 THEN X=3
260 IF B=2 THEN X=4
270 IF B=3 THEN X=5
280 .....
```

Als

```
200 IF A=1 AND B=1 THEN X=0
210 IF A=1 AND B=2 THEN X=1
220 IF A=1 AND B=3 THEN X=2
230 IF A<>1 AND B=1 THEN X=3
240 IF A<>1 AND B=2 THEN X=4
250 IF A<>1 AND B=3 THEN X=5
```

SCHLEIFEN

- Ganzzahlige Zaehler in FOR...NEXT-Schleifen benutzen, wo dies moeglich ist. Arithmetik mit ganzen Zahlen geht schneller als Arithmetik mit Zahlen einfacher und doppelter Genauigkeit.

- Die Variable in der Anweisung NEXT weglassen, wo dies moeglich ist. Wird eine Variable eingefuegt, benoetigt BASIC ein wenig Zeit, um zu pruefen, ob sie in Ordnung ist. Es kann noetig sein, eine Variable in der Anweisung NEXT anzufuegen, wenn aus geschachtelten Schleifen heraus verzweigt wird. Siehe unter Anweisung FOR und NEXT in Kapitel 4.

- FOR...NEXT Schleifen statt Kombinationen aus Anweisung IF und GOTO benutzen. Beispiel:

Besser

```
200 FOR I=1 TO 10
:
:
300 NEXT I
```

Als

```
200 I=1
210 .....
290 I=I+1
300 IF I<=10 THEN 210
```

- Unnoetigen Code aus Schleifen entfernen.
Das sind Anweisungen, die die Schleife nicht beeinflussen, sowie nicht ausfuehrbare Anweisungen, wie REM und DATA. Beispiel:

Besser

```
10 A=B+1
20 FOR X=1 TO 100
30 IF D(X)>A THEN D(X)=A
40 NEXT X
```

Als

```
10 FOR X=1 TO 100
20 A=B+1
30 IF D(X)>A THEN D(X)=A
40 NEXT X
```

Im vorherigen Beispiel ist es nicht noetig, den Wert von A jedesmal zu berechnen, wenn die Schleife verarbeitet wird, da in der Schleife der Wert von A nie geaendert wird.

Im naechsten Beispiel wird eine nicht ausfuehrbare Anweisung gezeigt.

Besser

```
200 DATA 5,12,1943
210 FOR I=1 TO 100
:
:
300 NEXT I
```

Als

```
200 FOR I=1 TO 100
210 DATA 5,12,1943
:
:
300 NEXT I
```